Petra Perner (Ed.)

Machine Learning and Data Mining in Pattern Recognition

16th International Conference on Machine Learning and Data Mining, MLDM 2020, Amsterdam, The Netherlands, July 20-21, 2020 Proceedings

ibai Publishing
www.ibai-publishing.org

Volume Editor

Petra Perner
FutureLab Artificial Intelligence_IBaI_II
Institute of Computer Vision and Applied Computer Sciences,
IBaI PF 30 11 14
04251 Leipzig
E-mail: pperner@ibai-institut.de

The German National Library listed this publication in the German National Bibliography.
Detailed bibliographical data can be downloaded from http://dnb.ddb.de.

ibai-publishing
Prof. Dr. Petra Perner
PF 30 11 38
04251 Leipzig, Germany
E-mail: info@ibai-publishing.org
http://www.ibai-publishing.org

P-ISSN 1864-9734
E-ISSN 2699-5220
ISBN 978-3-942952-75-0

16th International Conference on Machine Learning and Data Mining, MLDM 2020

July 20-21, 2020, Amsterdam, The Netherlands
www.mldm.de

Chair

Petra Perner
Institute of Computer Vision and Applied Computer Sciences, IBaI
Germany

Program Committee

Reneta Barneva	The State University of New York at Fredonia, USA
Michelangelo Ceci	Universtiy of Bari, Italy
Ireneusz Czarnowski	Gdynia Maritime University, Poland
Roberto Corrizo	Universtiy of Bari, Italy
Christoph F. Eick	Universtiy of Houston, USA
Mark J. Embrechts	Rensselaer Polytechnic Institute and CardioMag Imaging, Inc, USA
Ana Fred	Technical University of Lisboa, Portugal
Giorgio Giacinto	University of Cagliari, Italy
Aminata Kane	Concordia University, Canada
Piet Kommers	University of Twente, The Netherlands
Olga Krasotkina	Russian State University, Russia
Dimitris Karras	Chalkis Institute of Technology, Greece
Adam Krzyzak	Concordia University, Canada
Valerio Pascucci	University of Utah, USA
Gianvito Pio	University of Bari, Italy
Francis E.H. Tay	National University of Singapore, Singapore
Turki Turki	King Abdulaziz University, Saudi Arabia
Zeev Volkovich	ORT Braude College of Engineering, Israel
Patrick Wang	Northeastern University, USA

Preface

The pandemic "Corona" has put us this year before a difficult time. With care we have kept to the hygiene rules not to get an infection with the virus Covid-19. With mask we have got into coaches and trains, have made our purchases or on work worked. Home office was the catchword of these days. The universities and research facilities have maintained only a small emergency company and lectures were held as online lectures. From home we have tried to do our scientific works. In 1-to-1 tele-phone calls or phone conferences we have organized with our colleagues the work and have discussed important results of the research. Under it the efficiency suffers what is easy to understand.

In the beginning of the pandemic fell the deadline of our conference. Insecurity spread. The figures of the infected persons increased rapidly. The virus spreads out in more and more countries and was further carried by continent to continent. Soon stood the whole world in the spell of Corona. A conference was the last to this in this situation most thought.

In this situation appeared once again which high demands for a scientist are made. It belongs to the job of a scientist that he presents his scientific results in conferences and makes thus his results of a wide public immediately available. A scientist should have well organized his research, should be able to do his scientific tasks and duties in a flexible way, and should have financed his research with suitable financial means. Only those who were meeting these rules could successfully continue in their professional research work.

The best of the best of us are represented with their papers in this volume. They presented themselves personal or in online presentations in the conference. The acceptance rate for the submitted paper of our conference was 33% percent for long paper as well as short papers. Because of many refusals because of missing financial means or other reasons the acceptance rate decreased to few percent. This shows once more the excellent quality of these scientists. Their papers are of most excellent quality and expand the state-of-the-art in an excellent way. This proceeding volume present mostly theoretical work on known topics such as clustering, classification and prediction, graph mining but also application-oriented theoretical work for different purposes based deep learning and others. One invited talk was given by Prof. Dan A. Simovici on a theoretical subject on “Information theoretic approaches in data mining”. His paper is also included in the proceedings.

The proceedings will be freely accessible as an OPEN-ACCESS Proceedings of a wide public so that, the new acquired knowledge on the different subjects is able to spread around quickly worldwide. You can find the proceedings at http://www.ibai-publishing.org/html/proceeding2020.php.

In this time, flexibility was a must Because the situation in the USA was still difficult, we have moved the conference to Amsterdam in the Netherlands. Here a variety of the participants was able to do outward journeys. The ones who could not travel, were online present.

Extended versions of selected papers will appear in the international journal Transactions on Machine Learning and Data Mining (www.ibai-publishing.org/journal/mldm).

We invite you to join us in 2021 in New York again to the next International Conference on Machine Learning and Data Mining MLDM. The conference will run again under the umbrella of the Worldcongress (www.worldcongressdsa.com) "The Frontiers in Intelligent Data and Signal Analysis, DSA2020" that combines under his roof the following three events: International Conferences Machine Learning and Data Mining MLDM, the Industrial Conference on Data Mining ICDM , and the International Conference on Mass Data Analysis of Signals and Images in Artificial Intelligence and Pattern Recognition with Application in with Applications in Medicine, r/g/b Biotechnology, Food Industries and Dietetics, Biometry and Security, Agriculture, Drug Discover, and System Biology MDA-AI&PR.

We will give then the tutorials on Data Mining, Case-Based Reasoning, and Intelligent Image Analysis again (http://www.data-mining-forum.de/tutorials.php) again. The workshops running in connection with ICDM will also be given (http://www.data-mining-forum.de/workshops.php).

We would warmly invite you with pleasure to contribute to this conference. Please come and join us. We are awaiting you.

July, 2020 Petra Perner

Table of Content

Invited Talk

Regular Papers

Information-theoretical Approaches in Data Mining

Dan A. Simovici

University of Massachusetts Boston
Dan.Simovici@umb.edu

Abstract. The paper is centered on an algebraic approach to the notion of entropy and of several related concepts (conditional entropy, gain, metrics on sets of partitions produced by entropic approaches). Partitions are naturally associated with major data mining problems such as classification, clustering, data quality evaluations, and data preparation. This areas benefit from an algebraic and geometric study of metric structures defined on partitions. We discuss data mining techniques that use metrics defined on sets of partitions of finite sets such as decision tree induction, feature selection, and data discretization.

Keywords: partition · entropy · Gini index · conditional entropy · entropy gain · generalized entropic metric

1 Introduction

We present data mining techniques that use metrics defined on sets of partitions of finite sets derived from information-theoretical properties of partitions. Partitions are naturally associated with object attributes and major data mining problems such as classification, clustering, data quality evaluations, and data preparation benefit from an algebraic and geometric study of metric structures defined on partitions.

We begin in Section 1 by introducing fundamental facts that concern partitions of finite sets that will play a central role in this paper. The notion of entropy is discussed both from a probabilistic point of view and an algebraic approach in Section 2. The algebraic and computational aspects of entropy of partitions are the focus of our approach. In Section 3 we present the notions of conditional entropy and entropy gain. These notions allow us to define in Section 4 the entropic metric space of partitions of a finite sets and the the contingency matrix of two partitions. The remaining section of the paper are concerned with several applications of the notions studied up to this point.

Section 5 is dedicated to partitions defined on tables and decision trees where we present a generalization of de Mántaras metric algorithm. This is followed by an entropic algorithm for feature selection through clustering in Section 6, and a greedy algorithm for supervised discretization in Section 7. The final section proposes an entropic approach to the median partition problem and presents our conclusions.

Next, we discuss several mathematical preliminaries and notations. Let $S = \{x_1, \ldots, x_n\}$ be a finite set. A *partition* of S is a non-empty collection of non-empty subsets of S, $\pi = \{B_1, \ldots, B_m\}$ such that $B_i \cap B_j = \emptyset$ for $1 \leqslant i, j \leqslant m$ and $i \neq j$ and $\bigcup_{i=1}^{n} B_i = S$. The sets $B_1, \ldots, B_n$ are the *blocks* of π.

We denote by α_S the partition $\alpha_S = \{\{x_1\}, \ldots, \{x_n\}\}$ and by ω_S the partition that has a single block, $\omega_S = \{S\}$. The set of partitions on S is denoted by $\mathsf{PART}(S)$.

The set of partitions is equipped with a partial order. For $\pi, \sigma \in \mathsf{PART}(S)$ we write $\pi \leqslant \sigma$ if each block B of π is included in a block C of σ. If $\pi \leqslant \sigma$ it is easy to see that each block C of σ equals the union of the blocks of π included in C. Also, we have $\alpha_S \leqslant \pi \leqslant \omega_S$ for every $\pi \in \mathsf{PART}(S)$.

If $\pi \leqslant \sigma$ and there is no partition $\tau \in \mathsf{PART}(S)$ that is distinct from π and σ such that $\pi \leqslant \tau \leqslant \sigma$, then we say that σ *covers* τ and we write $\pi \lhd \sigma$. It is easy to see that if $\pi = \{B_1, \ldots, B_m\}$ and $\pi \lhd \sigma$, then there exist j, k such that $1 \leqslant j < k \leqslant m$ such that $\sigma = \{B_1, \ldots, B_{j-1}, B_{j+1}, \ldots, B_{k-1}, B_{k+1}, \ldots, B_j \cup B_k\}$. In other words, if $\pi \lhd \sigma$ then σ is obtained by fusing two of the blocks of π.

The greast lower bound in the poset $(\mathsf{PART}(S), \leqslant)$ of two partitions $\pi = \{B_1, \ldots, B_m\}$ and $\tau = \{D_1, \ldots, D_p\}$ in $\mathsf{PART}(S)$ is the partition $\pi \wedge \tau$

$$\pi \wedge \tau = \{B_i \cap D_j \mid B_i \in \pi, D_j \in \sigma \text{ and } B_i \cap D_j \neq \emptyset\}.$$

In the same poset, the least upper bound $\pi \vee \sigma$ has a more complicated description that can be found in [32].

2 Entropies for Probability Distributions and Partitions

The notion of *entropy* is a probabilistic concept that lies at the foundation of information theory. The use of entropy and several related is crucial in many machine learning areas: decision trees, clustering, maximum likelihood algorithms, etc. A powerful generalization of this notion was introduced in [16] and [9], and axiomatized in [29].

We will define entropy using an algebraic setting by introducing the notion of entropy of a partition of a finite set. This approach allows us to take advantage of the partial order that is naturally defined on the set of partitions. Actually, we introduce a generalization of the notion of entropy that has the *Gini index* and *Shannon entropy* as special cases.

In this paper we rely on elementary concepts from lattice theory that can be accessed using [4, 32].

In classical information theory the *Shannon entropy* of a probability distribution $\mathbf{p} = (p_1, \ldots, p_m)$, where $p_i > 0$ for $1 \leqslant i \leqslant m$ and $p_1 + \cdots + p_m = 1$ is defined as

$$\mathcal{H}(p_1, \ldots, p_m) = -\sum_{i=1}^{m} p_i \log_2 p_i = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}.$$

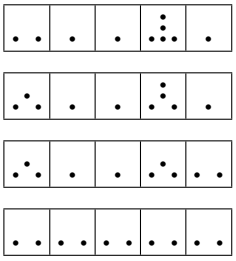
If $\pi = \{B_1, \ldots, B_m\}$ is a partition of a set S, then a probability distribution $\mathbf{p}_\pi$ can be defined as

$$\mathbf{p}_\pi = \left(\frac{|B_1|}{|S|}, \cdots, \frac{|B_m|}{|S|} \right).$$

Accordingly, we can define the *Shannon entropy of a partition* π as:

$$\mathcal{H}(\pi) = -\sum_{i=1}^{m} \frac{|B_i|}{|S|} \log_2 \frac{|B_i|}{|S|}.$$

Example 1. Let S be a set containing ten elements and let $\pi_1, \pi_2, \pi_3, \pi_4$ be the four partitions shown below.



The partition π_1, which is the most uniform (each block containing two elements), has the largest entropy. At the other end of the range, partition π_4 has a strong concentration of elements in its fourth block and the lowest entropy. Thus, the entropy can be viewed as a measure of impurity of a partition.

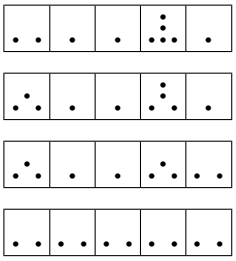
Definition 1. *The* Gini index of π *is the number*

$$\mathit{gini}(\pi) = 1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|S|} \right)^2.$$

Note that the gini-index of a partition is double the value of the generalized entropy $\mathcal{H}_2(\pi)$.

Like Shannon entropy, the Gini index can be used to evaluate the uniformity of the distribution of the elements of S in the blocks of π because both $\mathcal{H}(\pi)$ and $\mathsf{gini}(\pi)$ increase with the uniformity of the distribution of the elements of S.

Example 2. Results concerning the Gini index are shown next:



Definition 2. *Let $\pi = \{B_1, \ldots, B_m\}$ be a partition of a set S and let $\beta > 1$. The* β-entropy *of a partition π is the number*

$$\mathcal{H}_\beta(\pi) = \frac{1}{1 - 2^{1-\beta}} \cdot \left(1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|S|}\right)^\beta\right).$$

If $\beta = 2$, we obtain $\mathcal{H}_2(\pi)$, which is twice the Gini index,

$$\mathcal{H}_\beta(S, \pi) = 2 \cdot \left(1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|S|}\right)^2\right).$$

The *Gini index*, $\mathsf{gini}(\pi) = 1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|S|}\right)^2$, is widely used in machine learning and data mining.

When we take $\lim_{\beta \to 1} \mathcal{H}_\beta(\pi)$ we obtain the Shannon entropy! Indeed, we can write:

$$\begin{aligned}
\lim_{\beta \to 1} \mathcal{H}_\beta(\pi) &= \lim_{\beta \to 1} \frac{1}{1 - 2^{1-\beta}} \cdot \left(1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|S|}\right)^\beta\right) \\
&= \lim_{\beta \to 1} \frac{-\sum_{i=1}^{m} \left(\frac{|B_i|}{|S|}\right)^\beta \ln \frac{|B_i|}{|S|}}{2^{1-\beta} \ln 2} \\
&\quad \text{(by l'Hôpital Rule)} \\
&= -\sum_{i=1}^{m} \frac{|B_i|}{|S|} \log_2 \frac{|B_i|}{|S|}.
\end{aligned}$$

3 Conditional Entropy and Entropy Gain

Let π be a partition of the set S and let C be a non-empty subset of S.

The *trace* of π on C is the partition π_C of C given by:

$$\pi_C = \{B \cap C | B \in \pi \text{ such that } B \cap C \neq \emptyset\}.$$

Clearly, $\pi_C \in \mathsf{PART}(C)$; also, if C is a block of π, then $\pi_C = \omega_C$.

Definition 3. *Let $\pi, \sigma \in \mathsf{PART}(S)$ and let $\sigma = \{C_1, \ldots, C_n\}$. The* β-conditional entropy *of the partitions $\pi, \sigma \in \mathsf{PART}(S)$ is defined by*

$$\mathcal{H}_\beta(\pi|\sigma) = \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_j}).$$

The *Shannon conditional entropy* given by

$$\mathcal{H}(\pi|\sigma) = \sum_{j=1}^{n} \frac{|C_j|}{|S|} \mathcal{H}(\pi_{C_j}).$$

is a *limit case of the β-condional entropy*, that is, $\mathcal{H}(\pi|\sigma) = \lim_{\beta\to 1}\mathcal{H}_\beta(\pi|\sigma)$.

Note that for $\pi \in \mathsf{PART}(S)$ we have $\mathcal{H}_\beta(\pi|\omega_S) = \mathcal{H}_\beta(\pi)$ and that

$$\mathcal{H}_\beta(\omega_S|\sigma) = \sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\omega_{C_j}) = 0,$$
$$\mathcal{H}_\beta(\pi|\alpha_S) = \sum_{j=1}^{n}\frac{1}{|S|}\mathcal{H}(\pi_{\{x_j\}}) = 0$$

for every partition $\pi \in \mathsf{PART}(S)$.

For $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$, the conditional entropy can be written as:

$$\begin{aligned}\mathcal{H}_\beta(\pi|\sigma) &= \sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta \sum_{i=1}^{m}\frac{1}{1-2^{1-\beta}}\left[1-\left(\frac{|B_i\cap C_j|}{|C_j|}\right)^\beta\right]\\ &= \frac{1}{1-2^{1-\beta}}\sum_{j=1}^{n}\left(\left(\frac{|C_j|}{|S|}\right)^\beta - \sum_{i=1}^{m}\left(\frac{|B_i\cap C_j|}{|S|}\right)^\beta\right)\\ &= \mathcal{H}_\beta(\pi\wedge\sigma) - \mathcal{H}_\beta(\sigma). \end{aligned} \tag{1}$$

For the special case when $\pi = \alpha_S$, we can write

$$\mathcal{H}_\beta(\alpha_S|\sigma) = \sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\alpha_{C_j}) = \frac{1}{1-2^{1-\beta}}\left(\sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta - \frac{1}{|S|^{\beta-1}}\right). \tag{2}$$

If σ, π are two partitions of S and $\pi \geqslant \sigma$, then σ is more informative than π regarding the elements of S. This intuition is captured by the following statement.

Theorem 1. *Let S be a finite set and let $\pi, \sigma \in \mathsf{PART}(S)$. We have $\mathcal{H}_\beta(\pi|\sigma) = \mathcal{H}_\beta(\pi)$ if and only if $\pi \geqslant \sigma$.*

Proof. Suppose that $\sigma = \{C_1, \ldots, C_n\}$. If $\pi \geqslant \sigma$, each block of σ is included in a block of π and, therefore, we have $\pi_{C_j} = \omega_{C_j}$ for $1 \leqslant j \leqslant n$. Consequently, we have:

$$\mathcal{H}_\beta(\pi|\sigma) = \sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\omega_{C_j}) = 0.$$

Conversely, suppose that

$$\mathcal{H}_\beta(\pi|\sigma) = \sum_{j=1}^{n}\left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_j}) = 0.$$

This implies $\mathcal{H}_\beta(\pi_{C_j}) = 0$ for $1 \leqslant j \leqslant n$, which means that $\pi_{C_j} = \omega_{C_j}$ for $1 \leqslant j \leqslant n$ by a previous remark. This means that every block C_j of σ is included in a block of π, so $\pi \geqslant \sigma$.

Definition 4. *Let S, T be two disjoint sets and let $\sigma = \{B_1, \ldots, B_m\} \in PART(S)$ and $\tau = \{C_1, \ldots, C_n\} \in PART(T)$. The* sum of the partitions σ and τ is the partition $\pi + \sigma$ *of the set $S \cup T$ given by:*

$$\pi + \sigma = \{B_1, \ldots, B_m, C_1, \ldots, C_n\}.$$

Example 3. Let $\pi = \{B_1, B_2, B_3\} \in \mathsf{PART}(S)$ and let $\sigma = \{C_1, C_2, C_3, C_4\} \in \mathsf{PART}(T)$, where S and T are two disjoint sets (see Figure 1). The partition $\pi + \sigma \in \mathsf{PART}(S \cup T)$ is $\pi + \sigma = \{B_1, B_2, B_3, C_1, C_2, C_3, C_4\}$.

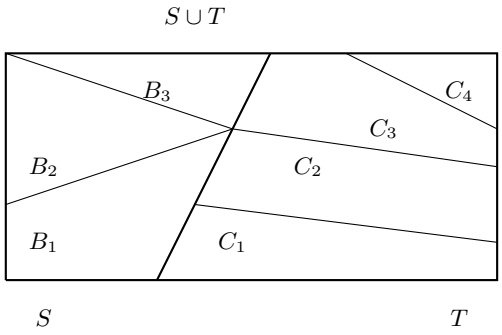


Fig. 1. The sum of partitions π and σ.

Example 4. Let $S = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ and let

$$\pi = \{\{x_1\}, \{x_2, x_3, x_4, x_5, x_6\}, \{x_7\}\}$$
$$\tau = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6, x_7\}\}.$$

We have

$$\pi \wedge \tau = \{\{x_1\}, \{x_2, x_3\}, \{x_4, x_5, x_6\}, \{x_7\}\}.$$

The next statement is a generalization of a well-known property of Shannon's entropy.

Theorem 2. *Let π and σ be two partitions of a finite set S. We have:*

$$\mathcal{H}_\beta(\pi \wedge \sigma) = \mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma) = \mathcal{H}_\beta(\sigma|\pi) + \mathcal{H}_\beta(\pi).$$

Proof. Let $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$ be two partitions of S. We have

$$\begin{aligned}
&\mathcal{H}_\beta(\pi \wedge \sigma) - \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_j}) \\
&= \frac{1}{1-2^{1-\beta}} \left(1 - \sum_i \sum_j \left(\frac{|B_i \cap C_j|}{|S|}\right)^\beta\right) \\
&\quad - \frac{1}{1-2^{1-\beta}} \sum_j \left(\frac{|C_j|}{|S|}\right)^\beta \left(1 - \sum_i \left(\frac{|B_i \cap C_j|}{|C_j|}\right)^\beta\right) = \mathcal{H}_\beta(\sigma).
\end{aligned}$$

From the result established above

$$\mathcal{H}_\beta(\pi \wedge \sigma) = \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_j}) + \mathcal{H}_\beta(\sigma),$$

we obtain

$$\mathcal{H}_\beta(\pi \wedge \sigma) = \mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma).$$

The second equality has a similar proof.

Definition 5. *Let π, σ be two partitions of a set S, where $\sigma = \{C_1, \ldots, C_n\}$.*

The β-information gain *of σ is the number*

$$\mathit{gain}_\beta(\pi, \sigma) = \mathcal{H}_\beta(\pi) - \mathcal{H}_\beta(\pi|\sigma).$$

Let $\pi \in \mathsf{PART}(S)$, and let $\sigma = \{C_1, \ldots, C_n\} \in \mathsf{PART}(S)$. If $\beta \to 1$ the *information gain of the Shannon entropy* is

$$\begin{aligned}
\mathsf{gain}(\pi, \sigma) &= \mathcal{H}(\pi) - \mathcal{H}(\pi|\sigma) \\
&= \mathcal{H}(\pi) - \sum_{i=1}^{n} \frac{|C_j|}{|S|} \mathcal{H}(\pi_{C_j}).
\end{aligned}$$

Note that $\pi \geqslant \sigma$, where $\pi, \sigma \in \mathsf{PART}(S)$, we have $\mathcal{H}_\beta(\pi|\sigma) = \mathcal{H}_\beta(\pi)$ if and only if

$$\mathsf{gain}_\beta(\pi, \sigma) = \mathcal{H}_\beta(\pi) - \mathcal{H}_\beta(\pi|\sigma) = 0.$$

4 The Entropic Metric Space of Partitions of Finite Sets

Theorem 3. *Let A, B be two finite, disjoint sets and let $\pi = \{B_1, \ldots, B_m\} \in \mathit{PART}(A)$, $\sigma = \{C_1, \ldots, C_n\} \in \mathit{PART}(B)$. We have:*

$$|A|^\beta \mathcal{H}_\beta(\pi) + |B|^\beta \mathcal{H}_\beta(\sigma) \leqslant (|A| + |B|)^\beta \mathcal{H}_\beta(\pi + \sigma).$$

Proof. The left member of the inequality of the theorem can be rewritten as:

$$\begin{aligned}
&|A|^\beta \mathcal{H}_\beta(\pi) + |B|^\beta \mathcal{H}_\beta(\sigma) \\
&= |A|^\beta \frac{1}{1-2^{\beta-1}} \left(1 - \sum_{i=1}^{m} \left(\frac{|B_i|}{|A|}\right)^\beta\right) \\
&\quad + |B|^\beta \frac{1}{1-2^{\beta-1}} \left(1 - \sum_{j=1}^{n} \left(\frac{|C_j|}{|B|}\right)^\beta\right) \\
&= \frac{1}{1-2^{\beta-1}} \left(|A|^\beta + |B|^\beta - \sum_{i=1}^{m} |B_i|^\beta - \sum_{j=1}^{n} |C_j|^\beta\right).
\end{aligned}$$

The right hand member of the same inequality is:

$$\begin{aligned}
&(|A| + |B|)^\beta \mathcal{H}_\beta(\pi + \sigma) \\
&= \frac{1}{1-2^{\beta-1}} \left((|A| + |B|)^\beta - \sum_{i=1}^{m} |B_i|^\beta - \sum_{j=1}^{n} |C_j|^\beta\right).
\end{aligned}$$

The inequality of the theorem follows from the fact that

$$|A|^\beta + |B|^\beta \leqslant (|A| + |B|)^\beta$$

because $\beta > 1$.

Theorem 4. *Let $\pi, \sigma \in \mathsf{PART}(S)$, where S is a finite set. We have $\mathcal{H}_\beta(\pi|\sigma) = 0$ if and only if $\sigma \leqslant \pi$.*

Proof. Suppose that $\sigma = \{C_1, \ldots, C_n\}$. If $\sigma \leqslant \pi$, then $\pi_{C_j} = \omega_{C_j}$ for $1 \leqslant j \leqslant n$ and, therefore,

$$\mathcal{H}_\beta(\pi|\sigma) = \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\omega_{C_j}) = 0.$$

Conversely, if $\mathcal{H}_\beta(\pi|\sigma) = \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|}\right)^\beta \mathcal{H}_\beta(\omega_{C_j}) = 0$, it follows that $\mathcal{H}_\beta(\pi_{C_j}) = 0$ for $1 \leqslant j \leqslant n$, which means that $\pi_{C_j} = \omega_{C_j}$ for $1 \leqslant j \leqslant n$. This means that every block C_j of σ is included in a block of π, so $\sigma \leqslant \pi$.

Corollary 1. *Let $\pi, \sigma \in \mathsf{PART}(S)$, where S is a finite set. We have $\mathcal{H}_\beta(\pi \wedge \sigma) = \mathcal{H}_\beta(\pi)$ if and only if $\pi \leqslant \sigma$.*

Proof. Since

$$\mathcal{H}_\beta(\pi \wedge \sigma) = \mathcal{H}_\beta(\sigma|\pi) + \mathcal{H}_\beta(\pi)$$

the equality of the corollary is equivalent to $\mathcal{H}_\beta(\sigma|\pi) = 0$, which is equivalent to $\pi \leqslant \sigma$ by Theorem 4.

Theorem 5. *Let $\pi, \pi', \sigma, \sigma' \in \mathsf{PART}(S)$. The β-conditional entropy $\mathcal{H}_\beta(\pi|\sigma)$ is dually monotonic relative to the first argument and monotonic with respect to the second argument. In other words, if $\pi \leqslant \pi'$, then $\mathcal{H}_\beta(\pi|\sigma) \geqslant \mathcal{H}_\beta(\pi'|\sigma)$.*

Also, if $\sigma \leqslant \sigma'$ we have $\mathcal{H}_\beta(\pi|\sigma) \leqslant \mathcal{H}_\beta(\pi|\sigma')$.

Proof. If $\pi \leqslant \pi'$, we have $\pi \wedge \sigma \leqslant \pi' \wedge \sigma$. Therefore, $\mathcal{H}_\beta(\pi \wedge \sigma) \geqslant \mathcal{H}_\beta(\pi' \wedge \sigma)$. Thus, we have:

$$\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma) \geqslant \mathcal{H}_\beta(\pi'|\sigma) + \mathcal{H}_\beta(\sigma),$$

hence $\mathcal{H}_\beta(\pi|\sigma) \geqslant \mathcal{H}_\beta(\pi'|\sigma)$, which proves the dual monotonicity of $\mathcal{H}_\beta(\pi|\sigma)$ relative to its first argument.

To prove the monotonicity of $\mathcal{H}_\beta$ in its second argument, that is,

$$\sigma \leqslant \sigma' \text{ implies } \mathcal{H}_\beta(\pi|\sigma) \leqslant \mathcal{H}_\beta(\pi|\sigma'),$$

we can assume that σ is covered by σ', that is, $\sigma \leqslant \sigma'$.

If σ is not covered by σ', there exists a sequence of partitions:

$$\sigma = \tau_1 \lhd \tau_2 \lhd \cdots \lhd \tau_n = \sigma'$$

such that τ_i is covered by τ_{i+1} and the monotonicity will follow.

So, suppose that σ is covered by σ'. Then, we may assume that:

$$\sigma = \{C_1, \ldots, C_{n-2}, C_{n-1}, C_n\},$$

and

$$\sigma' = \{C_1, \ldots, C_{n-2}, C_{n-1} \cup C_n\}.$$

The inequality $\mathcal{H}_\beta(\pi|\sigma) \leqslant \mathcal{H}_\beta(\pi|\sigma')$, which amounts to

$$\begin{aligned}&\left(\frac{|C_{n-1}|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_{n-1}}) + \left(\frac{|C_n|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_n})\\ &\leqslant \left(\frac{|C_{n-1} \cup C_n|}{|S|}\right)^\beta \mathcal{H}_\beta(\pi_{C_{n-1} \cup C_n}).\end{aligned}$$

The result follows immediately from the fact that $\pi_{C_{n-1} \cup C_n} = \pi_{C_{n-1}} + \pi_{C_n}$.

Lemma 1. *Let π, σ, τ be three partitions of a finite set S. We have:*

$$\mathcal{H}_\beta(\pi|\sigma \wedge \tau) + \mathcal{H}_\beta(\sigma|\tau) = \mathcal{H}_\beta(\pi \wedge \sigma|\tau).$$

Proof. We have the equalities

$$\begin{aligned}\mathcal{H}_\beta(\pi|\sigma \wedge \tau) &= \mathcal{H}_\beta(\pi \wedge \sigma \wedge \tau) - \mathcal{H}_\beta(\sigma \wedge \tau)\\ \mathcal{H}_\beta(\sigma|\tau) &= \mathcal{H}_\beta(\sigma \wedge \tau) - \mathcal{H}_\beta(\tau).\end{aligned}$$

By adding these these equalities we obtain the desired equality.

Theorem 6. *Let π, σ, τ be three partitions of a finite set S. Then, we have*

$$\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) \geqslant \mathcal{H}_\beta(\pi|\tau).$$

Proof. The monotonicity properties of $\mathcal{H}_\beta$ allow us to write:

$$\begin{aligned}\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) &\geqslant \mathcal{H}_\beta(\pi|\sigma \wedge \tau) + \mathcal{H}_\beta(\sigma|\tau)\\ &= \mathcal{H}_\beta(\pi \wedge \sigma|\tau) \geqslant \mathcal{H}_\beta(\pi|\tau)\end{aligned}$$

by Lemma 1.

Theorem 7. *Let S be a finite set and let $d_\beta : \mathit{PART}(S) \times \mathit{PART}(S)\mathbb{R}_{\geqslant 0}$ be defined by*

$$d_\beta(\pi, \sigma) = \mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\pi)$$

for $\pi, \sigma \in \mathit{PART}(S)$. Then, d_β is a metric on $\mathit{PART}(S)$.

Proof. It is immediate that $d_\beta(\pi, \sigma) = 0$ if and only if $\pi = \sigma$ and that $d_\beta(\pi, \sigma) = d_\beta(\sigma, \pi)$. So, we need to prove only the triangular inequality:

$$d_\beta(\pi, \sigma) \leqslant d_\beta(\pi, \tau) + d_\beta(\tau, \sigma)$$

for all $\pi, \sigma, \tau \in \mathsf{PART}(S)$.

By the Theorem 6, it follows that

$$\begin{aligned}\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) &\geqslant \mathcal{H}_\beta(\pi|\tau),\\ \mathcal{H}_\beta(\sigma|\pi) + \mathcal{H}_\beta(\tau|\sigma) &\geqslant \mathcal{H}_\beta(\tau|\pi).\end{aligned}$$

Adding these equalities yields

$$d_\beta(\pi, \sigma) + d_\beta(\sigma, \tau) \geqslant d_\beta(\pi, \tau),$$

which means that d_β satisfies the triangular inequality.

Corollary 2. *If $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$ are two partitions of the set S, then*

$$d_\beta(\pi, \sigma) = \frac{1}{1 - 2^{1-\beta}} \left(\sum_{i=1}^{m} \left(\frac{|B_i|}{|S|} \right)^\beta + \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|} \right)^\beta - 2 \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\frac{|B_i \cap C_j|}{|S|} \right)^\beta \right).$$

Proof. The equality of the corollary follows immediately from Equality (1).

Taking $\beta = 2$ we obtain the distance

$$\begin{aligned} d_2(\pi, \sigma) &= 2 \left(\sum_{i=1}^{m} \left(\frac{|B_i|}{|S|} \right)^2 + \sum_{j=1}^{n} \left(\frac{|C_j|}{|S|} \right)^2 - 2 \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\frac{|B_i \cap C_j|}{|S|} \right)^2 \right)\\ &= \frac{2}{|S|^2} \left(\sum_{i=1}^{m} |B_i|^2 + \sum_{j=1}^{n} |C_j|^2 - 2 \sum_{i=1}^{m} \sum_{j=1}^{n} |B_i \cap C_j|^2 \right). \end{aligned} \tag{3}$$

Note that $d_\beta(\alpha_S, \pi) = \mathcal{H}_\beta(\alpha_S) - \mathcal{H}_\beta(\pi)$ and $d_\beta(\omega_S, \pi) = \mathcal{H}_\beta(\pi)$.

The metric d_β can be normalized, by defining the mapping $e_\beta : \mathsf{PART}(S)^2 \longrightarrow \mathbb{R}^{\geqslant 0}$ as

$$e_\beta(\pi, \sigma) = \frac{d_\beta(\pi, \sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma)}.$$

The next theorem is a generalization of a result of de Mántaras.

Theorem 8. *The mapping e_β is a metric on the set $\mathsf{PART}(S)$ and $0 \leqslant e_\beta(\pi,\sigma) \leqslant 1$ for every $\beta \geqslant 1$.*

Proof. We shall prove that e_β satisfies the triangular axiom, that is,

$$\frac{d_\beta(\pi,\sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma)} + \frac{d_\beta(\sigma,\tau)}{\mathcal{H}_\beta(\sigma \wedge \tau)} \geqslant \frac{d_\beta(\pi,\tau)}{\mathcal{H}_\beta(\pi \wedge \tau)}.$$

We begin by showing that

$$\frac{\mathcal{H}_\beta(\pi|\sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma)} + \frac{\mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\sigma \wedge \tau)} \geqslant \frac{\mathcal{H}_\beta(\pi|\tau)}{\mathcal{H}_\beta(\pi \wedge \tau)}.$$

By Theorem 2 we can write:

$$\begin{aligned}
&\frac{\mathcal{H}_\beta(\pi|\sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma)} + \frac{\mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\sigma \wedge \tau)} \\
&= \frac{\mathcal{H}_\beta(\pi|\sigma)}{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma)} + \frac{\mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\sigma|\tau) + \mathcal{H}_\beta(\tau)} \\
&\geqslant \frac{\mathcal{H}_\beta(\pi|\sigma)}{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) + \mathcal{H}_\beta(\tau)} + \frac{\mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) + \mathcal{H}_\beta(\tau)} \\
&= \frac{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\tau) + \mathcal{H}_\beta(\tau)} \\
&\geqslant \frac{\mathcal{H}_\beta(\pi|\tau)}{\mathcal{H}_\beta(\pi|\tau) + \mathcal{H}_\beta(\tau)} = \frac{\mathcal{H}_\beta(\pi|\tau)}{\mathcal{H}_\beta(\pi \wedge \tau)}.
\end{aligned}$$

Similarly, we have

$$\frac{\mathcal{H}_\beta(\sigma|\pi)}{\mathcal{H}_\beta(\pi \wedge \sigma)} + \frac{\mathcal{H}_\beta(\pi|\tau)}{\mathcal{H}_\beta(\pi \wedge \tau)} \geqslant \frac{\mathcal{H}_\beta(\sigma|\tau)}{\mathcal{H}_\beta(\sigma \wedge \tau)}.$$

By adding up the last inequality we obtain the triangular inequality for e_β.

Since $\mathcal{H}_\beta(\sigma|\pi) \leqslant \mathcal{H}_\beta(\sigma)$, we have $\mathcal{H}_\beta(\pi \wedge \sigma) \leqslant \mathcal{H}_\beta(\sigma) + \mathcal{H}_\beta(\pi)$. The inequalities $\mathcal{H}_\beta(\pi \wedge \sigma) \geqslant \mathcal{H}_\beta(\pi)$ and $\mathcal{H}_\beta(\pi \wedge \sigma) \geqslant \mathcal{H}_\beta(\sigma)$ imply

$$\frac{2\mathcal{H}_\beta(\pi \wedge \sigma)}{\mathcal{H}_\beta(\pi) + \mathcal{H}_\beta(\sigma)} \geqslant 1.$$

Therefore, we have

$$\begin{aligned}
e_\beta(\pi,\sigma) &= \frac{d_\beta(\pi,\sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma)} \\
&= \frac{\mathcal{H}_\beta(\pi|\sigma) + \mathcal{H}_\beta(\sigma|\pi)}{\mathcal{H}_\beta(\pi \wedge \sigma)} \\
&= 2 - \frac{\mathcal{H}_\beta(\pi) + \mathcal{H}_\beta(\sigma)}{\mathcal{H}_\beta(\pi \wedge \sigma} \in [0,1].
\end{aligned}$$

In [31] we studied valuations an lattices and, in particular, on the lattice of partitions of a finite set. Let $v_\beta : \mathsf{PART}(S) \longrightarrow \mathbb{R}_{\geqslant 0}$ be the valuation defined on the lattice of partitions of a set S as

$$v_\beta(\pi) = \sum_{i=1}^{m} |B_i|^\beta,$$

where $\pi = \{B_1, \ldots, B_m\} \in \mathsf{PART}(S)$. We have:

$$v_\beta(\pi) = |S|^\beta \left[1 - (1 - 2^{1-\beta})\mathcal{H}_\beta(\pi)\right], \tag{4}$$

and this equality implies that v_β is a monotonic function of partitions because $\mathcal{H}_\beta$ is an antimonotonic function of partitions.

Theorem 9. *The function $v_\beta : \mathsf{PART}(S) \longrightarrow \mathbb{R}_{\geqslant 0}$ satisfies the inequality:*

$$v_\beta(\pi \vee \sigma) + v_\beta(\pi \wedge \sigma) \geqslant v_\beta(\pi) + v_\beta(\sigma)$$

for every $\pi, \sigma \in \mathsf{PART}(S)$.

Proof. By Equality (1) we have

$$\mathcal{H}_\beta(\pi|\sigma) = \mathcal{H}_\beta(\pi \wedge \sigma) - \mathcal{H}_\beta(\sigma).$$

Since $\mathcal{H}_\beta$ is monotonic in its second argument (Theorem 5), it follows that $\sigma_1 \leqslant \sigma_2$ implies $\mathcal{H}_\beta(\pi|\sigma_1) \leqslant \mathcal{H}_\beta(\pi|\sigma_2)$ or, equivalently,

$$\mathcal{H}_\beta(\pi \wedge \sigma_1) + \mathcal{H}_\beta(\sigma_2) \leqslant \mathcal{H}_\beta(\pi \wedge \sigma_2) + \mathcal{H}_\beta(\sigma_1).$$

Choosing $\sigma_2 = \pi \vee \sigma_1$ we have

$$\mathcal{H}_\beta(\pi \wedge \sigma_1) + \mathcal{H}_\beta(\pi \vee \sigma_1) \leqslant \mathcal{H}_\beta(\pi \wedge (\pi \vee \sigma_1)) + \mathcal{H}_\beta(\sigma_1) = \mathcal{H}_\beta(\pi) + \mathcal{H}_\beta(\sigma_1),$$

by the absorption property of lattices. Equality (4) implies the inequality of the theorem.

It is interesting that $d_2(\pi, \beta)$, that is, the metric that correponds to the Gini entropy has been introduced previously starting from combinatorial considerations and is also known as the *Barthélemy-Montjardet* metric [2, 3, 26]. Namely, for a set S and two partitions $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$, the number of unordered pairs $\{x, y\}$ of elements of S that belong to the same block of π is

$$\sum_{i=1}^{m} \binom{|B_i|}{2} = \sum_{i=1}^{m} \frac{1}{2}(|B_i|^2 - |B_i|) = \frac{1}{2}\sum_{i=1}^{m} |B_i|^2 - \frac{1}{2}|S|.$$

The similar number for σ is $\frac{1}{2}\sum_{j=1}^{n} |C_j|^2 - \frac{1}{2}|S|$. For the partition $\pi \wedge \sigma$, the number of pairs that belong to the same block of both π and σ is $\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} |B_i \cap$

$C_j|^2 - \frac{1}{2}|S|$. Thus, the number of pairs that belong to exactly one block of either partition is

$$\frac{1}{2}\sum_{i=1}^{m}|B_i|^2 + \frac{1}{2}\sum_{j=1}^{n}|C_j|^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}|B_i \cap C_j|^2$$
$$= \frac{1}{2}\left(\sum_{i=1}^{m}|B_i|^2 + \sum_{j=1}^{n}|C_j|^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n}|B_i \cap C_j|^2\right) = \frac{1}{4}d_2(\pi,\sigma),$$

which shows that the Barthélemy-Montjardet metric is proportional with $d_2(\pi,\sigma)$.

A partition $\pi = \{B_1, \ldots, B_m\}$ of a set $S = \{x_1, \ldots, x_n\}$ can be represented as a sequence $\mathbf{v}_\pi$ of length n of numbers in $\{1, \ldots, m\}$, where $(\mathbf{v}_\pi)_j = k$ if $x_j \in B_k$ for $1 \leqslant j \leqslant n$ and $k \in \{1, \ldots, m\}$.

Let $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$ be two partitions of a finite set S. Their *contingency matrix* is the $(m \times n)$-matrix $M_{\pi,\sigma} = (m_{ij})$ defined as

$$m_{ij} = |B_i \cap C_j|$$

for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. The contingency matrix is easy to compute and many environments contain special tools for this computation.

Example 5. Let $S = \{x_1, \ldots, x_{12}\}$ be a set and let π, σ be the partitions defined as

$$\pi = \{\{x_1, x_4, x_7\}, \{x_2, x_5\}, \{x_3, x_8\}, \{x_6, x_9, x_{10}\}\},$$

and

$$\sigma = \{\{x_1, x_7, x_9\}, \{x_2, x_3, x_4\}, \{x_5, x_6, x_8, x_{10}\}\}.$$

These partition are represented by vectors of dimension $|S|$ that indicate the number of the block to which successive elements belong:

$$\mathbf{v}_\pi = (1, 2, 3, 1, 2, 4, 1, 3, 4, 4),$$
$$\mathbf{v}_\sigma = (1, 2, 2, 2, 3, 3, 1, 3, 1, 3).$$

If m_{ij} is the (i,j)-entry of the contingency matrix $M_{\pi,\sigma}$, the *Pearson* $\chi^2_{\pi,\sigma}$ *association index* of π and σ can be written in our framework as

$$\chi^2_{\pi,\sigma} = \sum_i \sum_j \frac{(p_{ij} - |B_i||C_j|)^2}{|B_i||C_j|}. \tag{5}$$

It is well-known (see [1]) that the asymptotic distribution of $\chi^2_{\pi,\sigma}$ is a χ^2-distribution with $|\pi|\ |\sigma|$ degrees of freedom.

The **R** systems applies the function `table` to $\mathbf{v}_\pi$ and $\mathbf{v}_\sigma$ to compute $M_{\pi,\sigma}$:

```
vpi <- c(1,2,3,1,2,4,1,3,4,4)
vsigma <- c(1,2,2,2,3,3,1,3,1,3)
M <- table(vpi,vsigma)
```

resulting in

```
   vsigma
vpi 1 2 3
  1 2 1 0
  2 0 1 1
  3 0 1 1
  4 1 0 2
```

For a partition $\pi \in \mathsf{PART}(\{1, \ldots, 10\})$ defined by

$$\pi = \{\{x_1, x_4, x_7\}, \{x_2, x_5\}, \{x_3, x_8\}, \{x_6, x_9, x_{10}\}\},$$

we have $\mathbf{v}_\pi = (1, 2, 3, 1, 2, 4, 1, 3, 4, 4)$; the sizes of the blocks of π are the components of the vector `blocksize` computed by the next function.

```
> blocksize <- function(v){
                 ss <- length(v)
                 mx <- max(v)
                 B <- integer(mx)
                 for (i in 1:mx){
                     B[i] = sum(v==i)
                 }
                 return (B)
                }
```

Thus, writing `blocksize(vpi)` results in

```
[1] 3 2 2 3
```

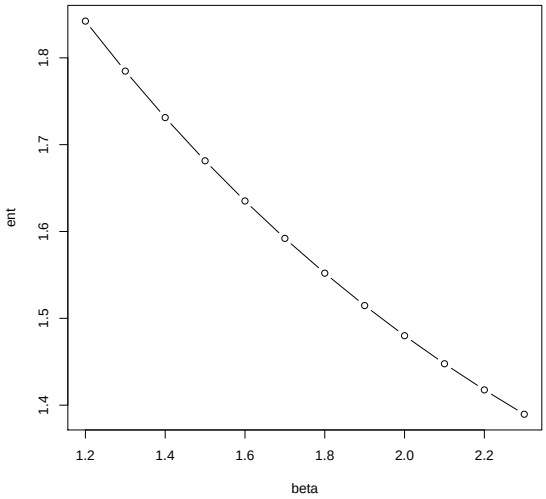
The β-entropy of the partition π is computed by the function `genent(v,beta)` given next:

```
genent <- function(v,beta) {
            ss <- length(v)
            blk <- blocksize(v)
            gent <- 1/(1- 2^(1-beta)) * (1 - sum((blk/ss)^beta))
            return(gent)
}
```

The code

```
plot(beta,genent(vpi,beta),type='b')
```

will generate the graph



5 Partitions Defined on Tables and Decision Trees

Let $H = \{A_1, \ldots, A_m\}$ be a set of symbols named *attributes*. To each attribute A a set containing at least two elements is attached named the *domain* of A and denoted by $\mathsf{Dom}(A)$. A *table content* is a sequence $S = (t_1, \ldots, t_n)$, where each t_i is a function $t_i : H \longrightarrow \bigcup_{j=1}^{m} \mathsf{Dom}(A_j)$ such that $t_i(A_j) \in \mathsf{Dom}(A_j)$ for each t_i that occurs in S. The components of S are referred to as *rows*.

A *table* is a pair $T = (H, S)$, where H is the *heading* and S is the content of the table. For t_i in c, the element $t_i(A)$ is the value of the tuple t_i on A. The set of values that occur under an attribute in a table T constitute the *active domain* of the attribute A denoted by $\mathsf{adom}(A)$.

If U is a subset of the set of attributes H and t is a row of the table, we refer to the restriction of t to the set U as the *projection* of t on the set U.

Each set of attributes U of a table $T = (H, S)$ partitions the rows of a table into blocks of rows that have equal values for that attribute. The resulting partition is denoted by π^U. Note that if U, V are two sets of attributes of T, then

$$\pi^{U \cup V} = \pi^U \wedge \pi^V. \tag{6}$$

In a study focused on decision trees [28] Quinlan introduced an inductive learning algorithm for constructing a decision tree. The proposed algorithm starts with a table having categorical attributes and performs a heuristic hill-climbing search without backtracking through the space of possible decision trees.

For each non-terminal node of the tree attribute is recursively selected a branch is created for each value of the attribute. The selection of attributes is

based on the information gain for each attribute, and the attribute that maximizes this gain is selected. A very lucid description of the Quinlan's algorithm can be also found in [25].

The induction of a decision tree starts with a table equipped with a special attribute D referred to as the *decision attribute*. The goal of Quilan's algorithm is to partition the table recursively splitting the content of the table based on the remaining attributes such that the leafs of the resulting tree contain pure (or almost pure) sets of tuples, that is, sets of tuples that have the same D-component.

The choice of the splitting attribute A is based on the information gain $\mathsf{gain}(\pi^D, \pi^A) = \mathcal{H}(\pi^D) - \mathcal{H}(\pi^D|\pi^A)$ and the splitting attribute is the one that maximizes $\mathsf{gain}(\pi^D, \pi^A)$.

Note that $\mathsf{gain}(\pi^D, \pi^A)$ is anti-monotonic relative to π^A because $\mathcal{H}(\pi^D|\pi^A)$ is monotonic in π^A. In other words, partitions that consist of numerous small blocks produce large gains, which shows that this measure is biased towards selecting attributes with many values. This results in trees that fragment excessively the data sets and produce complicated decision processes. This problem lead to the the introduction of the notion of gain ratio

$$\mathsf{gainratio}(\pi^D, \pi^A) = \frac{\mathsf{gain}(\pi^D, \pi^A)}{\mathcal{H}(\pi^A)}$$

that corrects this bias, but it may choose attributes with very low $\mathcal{H}(\pi^A)$ rather than those with high gain. In [24, 15] R. L. de Mántaras introduced a distance between partitions as attribute selection measure showed that this distance is not biased towards many-valued attributes.

Our generalization of the metric introduced in [24] affords the investigators to tune de Mántaras algorithm by modifying the parameter β in order to ameliorate the quality decision trees. In [30] we tested our approach on a number of data sets from [5]. We included only the results shown in Figure 1 which are fairly typical. Decision trees were constructed using metrics d_β, where β varied between 0.25 and 2.50. Note that for $\beta = 1$ the metric algorithm coincides with the approach of de Mántaras. In all cases, accurracy was assessed through 10-fold cross-validation. We also built standard decision trees using the J48 technique of the well-known WEKA package [33], which yielded the following results:

Standard J4.8

Data Set	accuracy	size	no. of leaves
Audiology	77.88	54	32
Hepatitis	83.87	21	11
Primary-tumor	39.82	88	47
Vote	94.94	7	4

Experimental evidence shows that β can be adapted such that accuracy is comparable, or better than the standard algorithm. The size of the trees and the number of leaves show that the proposed approach to decision trees results

consistently in smaller trees with fewer leaves. Below, we show the results on for several UCI data sets.

Audiology

β	accuracy	size	leaves
2.50	53.54	53	36
2.25	54.42	53	36
2.00	54.87	54	37
1.75	53.10	47	32
1.50	76.99	29	19
1.25	78.32	29	19
1.00	76.99	29	19
0.75	76.99	29	19
0.50	76.99	29	19
0.25	78.76	33	21

Hepatitis

β	accuracy	size	leaves
2.50	81.94	15	8
2.25	81.94	9	5
2.00	81.94	9	5
1.75	83.23	9	5
1.50	84.52	9	5
1.25	84.52	11	6
1.00	85.16	11	6
0.75	85.81	9	5
0.50	83.23	5	3
0.25	82.58	5	3

Primary-tumor

β	accuracy	size	leaves
2.50	34.81	50	28
2.25	35.99	31	17
2.00	37.76	33	18
1.75	36.28	29	16
1.50	41.89	40	22
1.25	42.18	38	21
1.00	42.48	81	45
0.75	41.30	48	27
0.50	43.36	62	35
0.25	44.25	56	32

Vote

β	accuracy	size	leaves
2.50	94.94	7	4
2.25	94.94	7	4
2.00	94.94	7	4
1.75	94.94	7	4
1.50	95.17	7	4
1.25	95.17	7	4
1.00	95.17	7	4
0.75	94.94	7	4
0.50	95.17	9	5
0.25	95.17	9	5

6 Feature Selection Through Clustering

The performance, robustness, and usefulness of classification algorithms are improved when relative few features are involved in the classification. Comprehensive survey on these issues are [13, 17]. Several approaches to feature selection have been explored including wrapper techniques [20], support vector machines [6], neural networks [19], and prototype-based feature selection [14] that is closed to our approach.

In [8] we presented an algorithm for feature selection that clusters attributes using an entropic metric and then makes use of the dendrogram of the resulting cluster hierarchy to choose the most relevant attributes. The main interest of this approach resides in the improved understanding of the structure of the analyzed data and of the relative importance of the attributes for the selection process.

A metric $\delta(A, B)$ on the set of attributes of a data set was defined in Section 5 starting with the metric between the corresponding partitions π^A, π^B. If we use the Gini metric, δ is defined as

$$\delta(\pi^A, \pi^B) = \sum_i |U_i|^2 + \sum_j |V_j|^2 - 2\sum_i \sum_j |U_i \cap V_j|^2,$$

where $\pi^A = \{U_1, \ldots, U_m\}$, and $\pi^B = \{V_1, \ldots, V_n\}$. The contingency matrix of these partitions is the matrix M_{π^A,π^B}. Let M_τ and m_τ be the largest and the smallest size of a block of the partition τ. For the Pearson association index $\chi^2_{\pi,\sigma}$ defined in Equality (5) we have the double inequality:

$$\frac{v(\pi)+v(\sigma)-\delta(\pi,\sigma)}{2M_\pi M_\sigma} - 2np + |S|^2 \leqslant \chi^2_{\pi,\sigma} \leqslant \frac{v(\pi)+v(\sigma)-\delta(\pi,\sigma)}{2m_\pi m_\sigma} - 2np + |S|^2,$$

where $v(\pi), v(\sigma)$ are the valuations defined in Equality (4). The above inequalities show that the Pearon coefficient $\chi^2_{\pi,\sigma}$ decreases with the distance between partitions and, thus, the probability that π and σ are independent increases with the distance between partitions. This suggests that partitions that are correlated are close in the sense of the *Barthélemy-Montjardet* metric. Therefore, if the attributes of the data set are clustered using the δ-distance between partitions we could replace clusters with their medoids and, thereby, drastically reduce the number of attributes involved in a classification without significant decreases in accuracy of the resulting classifiers. Experimental results described in [8] justify this claim. For example, an experiment conducted on the data set `votes` from the repository [5] which records the votes of 435 US congressman on 15 key questions (and each congressman is classified as a democrat or republican) shows that retaining only 7 key attribues as representatives of clusters obtained using the Ward clustering method does not result in an appreciable loss of accuracy. A classification produced with a wrapper with the `J48` algorithm results in an accuracy of 96.03%.

7 A Greedy Algorithm for Supervised Discretization

Frequently data sets have attributes with numerical domains which makes them unsuitable for certain data mining algorithms that deal mainly with nominal attributes, such as decision trees and naive Bayes classifiers. To use such algorithms we need to replace numerical attributes with nominal attributes that represent intervals of numerical domains with discrete values. This process, known to as *discretization*, has received a great deal of attention in the data mining literature and includes a variety of ideas ranging from fixed k-interval discretization [10], fuzzy discretization (see [21, 22]), Shannon- entropy discretization due to Fayyad and Irani presented in [12, 11], proportional k-interval discretization (see [35, 34]), or techniques that are capable of dealing with highly dependent attributes [23].

To discretize a numerical attribute B we select a sequence of numbers $t_1 < t_2 < \cdots < t_\ell$ in $\mathsf{adom}(B)$. Next, the attribute B is replaced by the nominal attribute $\hat{B}$ that has $\ell + 1$ distinct values in its active domain $\{k_0, k_1, \ldots, k_\ell\}$. Each B-component b of an object o is replaced by the discretized B-component k defined by

$$k = \begin{cases} k_0 & \text{if } b \leqslant t_1, \\ k_i & \text{if } t_i < b \leqslant t_{i+1} \text{ for } 1 \leqslant i \leqslant \ell - 1, \\ k_\ell & \text{if } t_\ell < b. \end{cases}$$

The starting point of our result is the observation that every nominal attribute A of a set of objects S induces a partition κ_A of the set S such that the objects t, s belong to the same block of the partition κ_A if their A-components are equal. Recall that SQL computes the partition κ_A using the `group by` option of a `select` phrase.

There are two types of discretization [33]: unsupervised discretization, where the discretization takes place without any knowledge of the classes to which objects belong, and supervised discretization which takes into account the classes of the objects. Our approach involves supervised discretization. Within our framework, to discretize a numerical attribute B amounts to constructing a partition of the active domain $\mathsf{adom}(B)$ taking into account the partition κ_A determined by the nominal class attribute A.

The set of numbers $\mathtt{T} = (t_1, t_2, \ldots, t_\ell)$ defines the discretization process and they will be referred to as class separators. The corresponding partition of $\mathsf{adom}(B)$ is $\pi_B^{\mathtt{T}} = \{Q_0, \ldots, Q_\ell\}$, where $Q_i = \{b \in \mathsf{Dom}(B) \mid t_i \leqslant b \leqslant t_{i+1}\}$ for $0 \leqslant i \leqslant \ell$, where $t_0 = -\infty$ and $t_{\ell+1} = \infty$. It is immediate that $\pi_B^{T \cup T'} = \pi_B^T \wedge \pi_B^{T'}$. The discretization process consists of replacing each value that falls in the block Q_i of $\pi_B^{\mathtt{T}}$ by i for $0 \leqslant i \leqslant \ell$.

Partitions of active attribute domains induce partitions of the set of objects. Namely, the partition of the set of objects S that corresponds to a partition π of $\mathsf{adom}(B)$, where B is a numerical attribute is denoted by π_*. A block of π_* consists of all objects whose B-component belong to the same block of π. Note that when $\pi = \alpha_{\mathsf{adom}(B)}$, then $\pi_* = \kappa_B$.

Suppose now that we have a numerical attribute B and a categorical attribute A that represents the class of each object $o_1, \ldots, o_\ell$, where

$$o_1[B] \leqslant o_2[B] \leqslant \ldots \leqslant o_n[B]. \tag{7}$$

Define the partition $\pi_{B,A}$ of $\mathsf{adom}(B)$ whose blocks consist of maximal subsequences $o_i[B], \ldots, o_\ell[B]$ of the sequence 7 such that every object o_j belongs to the same block of the partition π^A.

Example 6. Let $\{o_1, \ldots, o_9\}$ be a collection of nine objects sorted on the attribute B:

	$\cdots$	B	$\cdots$	A
o_1	$\cdots$	95.2	$\cdots$	Y
o_2	$\cdots$	110.1	$\cdots$	N
o_3	$\cdots$	120.0	$\cdots$	Y
o_4	$\cdots$	125.5	$\cdots$	Y
o_5	$\cdots$	130.1	$\cdots$	N
o_6	$\cdots$	140.0	$\cdots$	N
o_7	$\cdots$	140.5	$\cdots$	Y
o_8	$\cdots$	168.2	$\cdots$	Y
o_9	$\cdots$	190.5	$\cdots$	Y

The partition κ_A has two blocks $\{o_1, o_3, o_4, o_7, o_8, o_9\}$ and $\{o_2, o_5, 0_6\}$ corresponding to the values Y and N, respectively.

The partition $\pi_{B,A,*}$ is

$$\pi_{B,A,*} = \{\{o_1\}, \{o_2\}, \{o_3, o_4\}, \{o_5, o_6\}, \{o_7, o_8, o_9\}\}.$$

The blocks of this partition correspond to the longest subsequences of the sequence $(o_1, \ldots, o_9)$ that consists of objects that belong to the same A-class.

Fayyad [12, 11] showed that to obtain the least value of Shannon's conditional entropy $\mathcal{H}(pi_A|\pi^{\mathtt{T}}_{B*}$ the cutpoints t of $\mathtt{T}$ must be chosen among the boundary points of the partition $\pi_{B,A}$. This is a powerful result that limits drastically the number of cut points and improves the tractability of the discretization.

In [7] we proposed a generalization of Fayyad's discretization techniques that relies on the metric on partitions defined by β-entropy. We have shown that with an appropriate choice of the parameters of the discretization process the resulting decision trees are smaller, have fewer leaves, and display higher of accuracy as verified by stratified cross-validation. We have shown the following statement.

Theorem 10. *Let S be a collection of objects, where the class of an object is determined by the attribute A and let $\beta \in (1, 2]$. It $\mathtt{T}$ is a set of cutpoints such that the conditional entropy $\mathcal{H}_\beta(\kappa_A|\pi^{\mathtt{T}}_{B*})$ is minimal among the set of cutpoints with the same number of elements, then $\mathtt{T}$ consists of boundary points of the partition $\pi_{B,A}$ of $\mathsf{adom}(A)$.*

Based on Theorem 10 we developed and tested a new discretization algorithm in [7] on several machine learning data sets from UCI data sets [5] that have numerical attributes. After discretizations performed with several values of β (typically $\beta \in \{1.5, 1.8, 1.9, 2\}$) we built the decision trees on the discretized data sets using the WEKA J48 variant of C4.5 [33]. The discretization technique has a significant impact of the size and accuracy of the decision trees. The experimental results suggest that an appropriate choice of β can reduce significantly the size and number of leaves of the decision trees, roughly maintaining the accuracy (measured by stratified 5-fold cross validation) or even increasing the accuracy.

The size, number of leaves and accuracy of the trees are described in Table 1, where trees built using the Fayyad-Irani discretization method of J48 are designated as "standard".

8 Further Work and Conclusions

The notion of generalized entropy affords more flexibility in machine learning algorithm design. As we say, better decision trees, discretization techniques, and feature selection can benefit from using this generalization of Shannon entropy and the metrics associated with it.

A research problem that we propose is the computation of consensus partitions using entropic metrics. When several partitions $\pi_1, \ldots, \pi_n$ exist on a set S finding a *consensus partition* π aims to summarize these partitions. A general

Database	Experimental Results			
	Discretization method	Size	Number of leaves	Accuracy (stratified cross-validation)
heart-c	*standard*	*51*	*30*	*79.20*
	$\beta = 1.5$	20	14	77.36
	$\beta = 1.8$	28	18	77.36
	$\beta = 1.9$	35	22	76.01
	$\beta = 2.0$	54	32	76.01
glass	*standard*	*57*	*30*	*57.28*
	$\beta = 1.5$	32	24	71.02
	$\beta = 1.8$	56	50	77.10
	$\beta = 1.9$	64	58	67.57
	$\beta = 2.0$	92	82	66.35
ionosphere	*standard*	*35*	*18*	*90.88*
	$\beta = 1.5$	15	8	95.44
	$\beta = 1.8$	19	12	88.31
	$\beta = 1.9$	15	10	90.02
	$\beta = 2.0$	15	10	90.02
iris	*standard*	*9*	*5*	*95.33*
	$\beta = 1.5$	7	5	96
	$\beta = 1.8$	7	5	96
	$\beta = 1.9$	7	5	96
	$\beta = 2.0$	7	5	96
diabetes	*standard*	*43*	*22*	*74.08*
	$\beta = 1.8$	5	3	75.78
	$\beta = 1.9$	7	4	75.39
	$\beta = 2.0$	14	10	76.30

Table 1. Comparative Experimental Results for Decision Trees

approach proposed in [3] is to consider a metric d on $\mathsf{PART}(S)$ and to seek a partition π on S such that the sum $\sum_{i=1}^{n} d(\pi, \pi_i)$ is minimal. Entropic distances on $\mathsf{PART}(S)$ offer a natural background for solving this kind of problem.

The problem of finding a consensus partition for a finite collection of partitions was first proposed in [18] in the domain of social choice. Another area of interest for consensus partitions is in numerical taxonomy [27]. Finding a consensus partition [3] arises, for example, when consider several classifications that are regarded as approximations of a true classification that we need to recover, or when n partitions $\pi_t, \pi_{t+1}, \ldots, \pi_{t+n-1}$ result from measurements at times $t, t+1, \ldots, t+n-1$ and one seeks a smoothing consensus of these partitions.

The approach we propose entails the construction a chain of partitions $\sigma_0 \leqslant \sigma_1 \leqslant \cdots$, using an approach similar to hierarchical clustering starting with the partition $\pi_0 = \alpha_S$ and to seek σ in this chain such that the sum $\sum_{i=1}^{n} d_\beta(\sigma, \pi_i)$ is minimal.

References

1. Agresti, A.: An Introduction to Categorical Data Analysis. John Wiley, New York (1997)
2. Barthelemy, J.P.: Remarques sur les propriétés metriques des ensembles ordonnés. Mathématiques et Sciences humaines **61**, 39–60 (1978)
3. Barthélemy, J.P., Leclerc, B.: The median procedure for partitions. Partitioning data sets **19**, 3–34 (1995)
4. Birkhoff, G.: Lattice Theory. American Mathematical Society, Providence, third edn. (1995)
5. Blake, C.L., Merz, C.J.: Uci repository of machine learning databases (1998), university of California, Irvine, Dept. of Information and Computer Sciences
6. Brown, M., Grundy, W.N., Lin, D., Cristianini, N., Sugnet, C., Furey, T.S., Ares, M., Haussler, D.: Knowledge-based analysis of microarray gene expression data by using support vector machines. Proceedings of the National Academy of Sciences **97**(1), 262–267 (2000)
7. Butterworth, R., Simovici, D.A., Santos, G.S., Ohno-Machado, L.: A greedy algorithm for supervised discretization. Journal of biomedical informatics **37**(4), 285–292 (2004)
8. Butterworth, R., Piatetsky-Shapiro, G., Simovici, D.A.: On feature selection through clustering. In: Fifth IEEE International Conference on Data Mining (ICDM'05). pp. 4–pp. IEEE (2005)
9. Daróczy, Z.: Generalized information functions. Information and Control **16**, 35–51 (1970)
10. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proceedings of the 12th International Conference on Machine Learning, pp. 194–202. Elsevier (1995)
11. Fayyad, U.M.: On the induction of decision trees for multiple concept learning. Ph.D. thesis, U. of Michigan (1992)
12. Fayyad, U.M., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of the 12th Int. Joint Conference on Artificial Intelligence. pp. 1022–1027 (1993)

13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research **3**(Mar), 1157–1182 (2003)
14. Hanczar, B., Courtine, M., Benis, A., Hennegar, C., Clément, K., Zucker, J.D.: Improving classification of microarray data using prototype-based feature selection. ACM SIGKDD Explorations Newsletter **5**(2), 23–30 (2003)
15. Cerquides, J., de Mántaras, R.L.: Proposal and empirical comparison of a parallelizable distance-based discretization method. In: Proc. of the 3rd International Conference on Knowledge Discovery and Data Mining. pp. 139–142 (1997)
16. Havrda, J. H., Charvat, F. : Quantification methods of classification processes: Concepts of structural α-entropy. Kybernetica **3**, 30–35 (1967)
17. Jain, A., Zongker, D.: Feature selection: Evaluation, application, and small sample performance. IEEE transactions on pattern analysis and machine intelligence **19**(2), 153–158 (1997)
18. Kemeny, J.: Mathematics without numbers. Daedalus **88**, 577–591 (1959)
19. Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature medicine **7**(6), 673–679 (2001)
20. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial intelligence **97**(1-2), 273–324 (1997)
21. Kononenko, I.: Naive Bayes classifier and continuous attributes. Informatica **10**, 1–8 (1992)
22. Kononenko, I.: Inductive and Bayesian learning in medical diagnosis. Applied Artificial Intelligence **7**, 317–337 (1993)
23. M. Robnik, I.K.: Discretization of continuous attributes using relieff. In: Proc. of ERK-95. pp. 149–152 (1995)
24. de Mántaras, R.L.: A distance-based attribute selection measure for decision tree induction. Machine Learning **6**, 81–92 (1991)
25. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
26. Monjardet, B.: Metrics on partially ordered sets - a survey. Discrete mathematics **35**(1-3), 173–184 (1981)
27. Neumann, D.A., V. T. Norton, J.: Clustering anf isolation in the consensus problem. J. of Classification **3**, 281–297 (1986)
28. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
29. Simovici, D.A., Jaroszewicz, S.: An axiomatization of partition entropy. IEEE Transactions on Information Theory **48**(7), 2138–2142 (2002)
30. Simovici, D.A., Jaroszewicz, S.: A new metric splitting criterion for decision trees. The International Journal of Parallel, Emergent and Distributed Systems **21**(4), 239–256 (2006)
31. Simovici, D.A.: On submodular and supermodular functions on lattices and related structures. In: 2014 IEEE 44th International Symposium on Multiple-Valued Logic. pp. 202–207 (2014)
32. Simovici, D.A., Djeraba, C.: Mathematical Tools for Data Mining. Springer, London, second edn. (2014)
33. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco, fourth edn. (2016)
34. Yang, Y., Webb, G.I.: Weighted proportional k-interval discretization for naive-Bayes classifiers. In: Proceedings of PAKDD. pp. 501–512 (2003)

35. Yang, Y., Webb, G.I.: Proportional k-interval discretization for naive-Bayes classifiers. In: Proc. of the 12th European Conference on Machine Learning. pp. 564–575 (2001)

Identification of Urban Functional Areas Based on POI Cluster Analysis

Huifang Feng, Chong Jia, Zhenjuan Yang and Youji Xu

College of mathematics and statistics, Northwest Normal University, Lanzhou，China
hffeng@nwnu.edu.cn

Abstract. POI data contains abundant urban spatial information, which is an important carrier for the study of urban layout and urban functional areas analysis. In this paper, we propose an urban functional area recognition and analysis method based on POI data. It takes the Lanzhou City as a case study to analyze the main function and spatial distribution characteristics of the detailed functional areas. First, the map information of Lanzhou urban area is obtained by the BAIDU and the POI is matched with the urban map. Then, the urban area is partitioned into equal-size grid and the statistical characteristics of each POI categories in each grid are analyzed. Finally, we employ a partitioning around medoid clustering model to cluster the functional areas and propose the enrichment factor to detect the urban functional areas. The results show that the proposed algorithm can quickly and effectively identify the urban functional areas.

Keywords: Urban functional areas, Partitioning around medoid, Clustering algorithms, Point of interest, Lanzhou city

1 Introduction

Studying the pattern of urban functional areas is of great significance to the construction of smart cities and the implementation of long-term urban development goals.. However, the complexity of urban spatial structure and interaction mechanism make it very complex to in-depth study urban spatial structure. The traditional urban functional areas identification is mainly based on questionnaires and expert evaluation [1][2], which are more subjective. Although the spatial remote sensing technology can effectively realize the identification and division of urban functional areas [3], this method has the disadvantages of high cost of time-consuming and laborious manual interpretation and the inaccurate identification by machine-aided automatic interpretation methods.

POI (point of interest) data contains abundant urban spatial information, which including entities name, address, coordinates, type, etc. They have the advantages of detailed spatial information, huge data volume, and convenient access to reflect urban structure function. POI data is widely used in geospatial distribution and urban functional areas analysis. A reasonable layout of urban functional areas can improve the efficiency of urban land use, the living standards of urban residents, and promote

sustainable urban development. The POI data has the abundant urban spatial information and provides great potential for identifying and mapping urban functional areas. In this paper, we use POI data to study the identification of urban functional areas in a valley city Lanzhou. The major contributions are summarized as follows: we propose new identification method of urban functional area based on POI data. Taking Lanzhou, China as a case, we analyze the main function and spatial distribution characteristics of the detailed functional areas.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 provides the study area and POT data. Section 4 proposes new identification method of urban functional area based on POI data. Section 5 reports the results of analysis. Finally, the conclusion and our future work are presented in Section 6.

2 Related work

Based on 530 karaoke bars in Nanjing, Cui et al. analyzed the spatial distribution of these bars and the forming factors of their locations using the point pattern analysis method and cluster analysis method in GIS [4]. Xue et al. analyzed the similarity and difference between the POI data of residential and retail industries in Shenyang by kernel density analysis and statistical method [5]. Jia et al. used the nearest neighbor hierarchical clustering and spatial autocorrelation methods to study the spatial layout of public leisure facilities in Urumqi [6]. Based on the POI data of the catering industry in Nanjing, Zhu et al. explored the spatial pattern and influencing factors of the catering industry facilities in Nanjing by kernel density estimation and nearest neighbor analysis [7]. Wang et al. used the POI data of Tianjin to classify commercial interest points near subway stations, and constructed a coupling coordination degree model to enhance the commercial space vitality of the subway station. They also provided suggestions for the location of the subway station [8].

In addition to the distribution characteristics of POI, the spatial layouts of urban functional areas were studied based on POI data. The commercial POI data of Wuhan was processed by standard deviation ellipse, kernel density analysis and nearest neighbor distance analysis, and the commercial center of Wuhan was identified [9]. Chen et al. analyzed the urban commercial center and retail industry cluster areas based on the POI data of Guangzhou commercial institutions by the kernel density method. They also analyzed the distribution characteristics of hotspot area [10]. Based on the Beijing POI data, Lin et al. extracted the commercial centers within the Sixth Ring Road of Beijing by kernel density analysis, and constructed the analysis of the level of the commercial center based on the concept of “function”, system clustering and central geography theory [11]. Based on the POI data of Changchun City, Hao et al. analyzed the spatial pattern and agglomeration characteristics of the commercial center by kernel density estimation, location entropy index and Ripley's K function [12]. Chi et al. reclassified the POI data of Wuhan, the single functional area and mixed functional area of the city were quantitatively identified, and visualized the identified functional area with RGB color addition [13]. Through the overall analysis

of the service industry in Zhengdong New District and the spatial layout clustering of POI data in different industries, the direction of industrial structure optimization in different functional areas of Zhengdong New District was proposed from the perspective of planning [14].

Other studies have focused on identification and classification urban functional areas combining POI data and other factors. Based on hierarchical semantic cognitive structure, Zhang et al. used very high-resolution satellite images and POI data to classify functional areas in Beijing. The results showed that the accuracy of proposed method was better than that of SVM and potential Dirichlet assignment [15]. Zhai et al. introduced a neighborhood scale method combining Place2vec and POI to extract and identify urban functional areas [16]. Compared with other probabilistic topic models, their method had higher accuracy.

3 Study Area and Data

3.1 Study Area and Grid Generation

Lanzhou, the capital of Gansu Province, is an important transportation hub in the northwestern region. There are high mountains in the north and south of Lanzhou. The Yellow River passes through the city from west to east. The urban form belongs to a kind of non-compact band structure and a typical river valley city. Due to the limitation of geographical conditions, Lanzhou has formed a geographical space pattern with long and narrow things and high west and low east. The study area includes Chengguan District, Qilihe District, Anning District and Xigu District.

In this paper, the study area is divided into a series of sub-areas by grid method. Let the POI data set be P, $P = \{P_1, P_2, P_3, \cdots, P_n\}$, where $P_i = (lon_i, lat_i, category_i)$, (lon_i, lat_i) is the location information of the i-th POI. They represent longitude and latitude, respectively. $category_i$ represents the type of POI. $lon_{\max}, lon_{\min}, lat_{\min}, lat_{\min}$ indicates the maximum and minimum values of latitude and longitude of the POI. Therefore, the study area is described as $R = [lon_{\max}, lon_{\min}] \times [lat_{\min}, lat_{\min}]$. The study area is partitioned into equal-size grid. Suppose the length and width of the grid are L and W, respectively. The number of grid is equal to pq, where $p = \text{int}(\frac{lon_{\max} - lon_{\min}}{L}), q = \text{int}(\frac{lat_{\max} - lat_{\min}}{W})$, int(•) is the integral function.

The grid division of areas should be neither too large nor too small. If the grid is too large, the functional area identification is inaccurate. On the contrary, if the grid is too small, the amount of POI data in one grid is too small. This also leads to misidentification. In this paper, the length and width of the grid are set to $500m$. By calculation, $p = 61$, $q = 29$. There are 1769 grids in study area.

3.2 POI Data

A POI may be a restaurant, an attraction, a station, etc. The POI is generally composed of name and coordinates. In this paper, more than 121, 103 POI for Lanzhou City are fetched via application programming interfaces (APIs) provided by Baidu Map Services in November 2016. Raw POI data usually contains abnormal errors caused by random and systematic errors. For further analysis, the data cleaning method for the raw POI data is used to remove the obvious errors. For example, we delete the POI outside the study area according to the scope of the study area. After reclassification, POI data is divided into 9 categories (top-level) based on their business services. They include finance, leisure, food, shopping, transportation, civic & enterprise, accommodation, and services.

Table 1. Classifications of POI data

Category	Labels
Finance	bank, atm, insurance company, accounting,etc.
Leisure	cinema, theatre, gym, ktv, playground, internet café, spa, barber, beauty salon, etc.
Food	restaurant, bakery, bar, cold drinking shop, cafe, tea, etc.
Shopping	bookstore, supermarkets, grocery, agricultural market, specialty stores, building materials market, clothing, etc.
Trans	train station, bus station, airport, subway station.
Civic & Enterprise	courthouse, lawyer, police, city hall, school, hospital, etc.
Public	park square, museum, library, tourist, temple, church, landmark, intersection
Accommodation	lodge, hotel, hostel, apartment, community.
Services	car repair, car wash, gas station, service area.

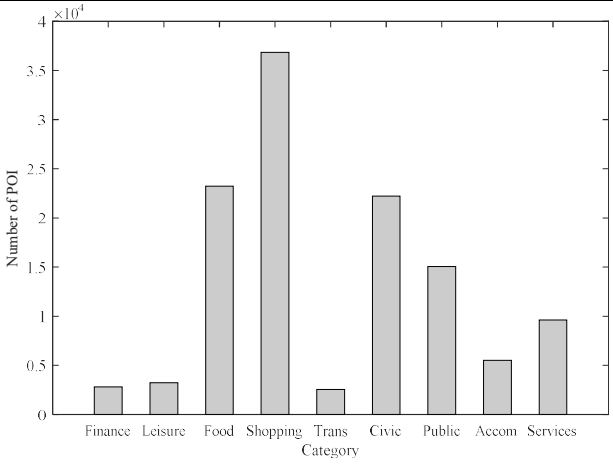


Fig. 1. The statistics of POI in Lanzhou

The detailed description of POI is shown in Table 1. The distribution of POI is shown in Figure 1. As can be seen from Figure 1, the proportion of food and shopping in the POI of Lanzhou City is large, and the proportion of Finance, leisure, trans, etc. is small, and the quantity difference is large.

4 Urban Functional Area Identification

4.1 PAM Clustering Algorithm

Cluster analysis is an unsupervised learning method and an important data mining algorithm. Popular clustering algorithms include k-means, Gaussian mixture models (GMMs), spectral clustering and hierarchical clustering, etc..[17] The k-medoids or partitioning around medoids (PAM) algorithm is a clustering algorithm similar to k-means algorithm [18]. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. The PAM algorithm is more robust than the k-means clustering algorithm and is insensitive to noise and outlier data. It can handle different types of data points and very effective for small data sets.

We first count the number of POIs of various types in each grid and construct data set D:

$$D = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \tag{1}$$

where n is the number of grids, p is the number of POI categories, here p=9. x_{ij} represents the number of j-th POI in the i-th grid. Then, the urban functional area identification is performed based on PAM clustering algorithm. The PAM clustering algorithm is shown in Table 2.

Table 2. PAM clustering algorithm based on POI data

Algorithm 1: PAM clustering algorithm based on POI data
Input: k : the number of clusters, D : POI data set Output: A set of k clusters .
(1) arbitrarily choose k objects in D as the initial cluster medoids.
(2) for each pair of non-selected object h and selected object i , calculate the total swapping cost T_{ih}
(3) for each pair of i and h If $T_{ih} < 0$, i is replaced by h then assign each non-selected object to the most similar representative object
(4) repeat steps 2 and 3 until no change happens.

In Algorithm 1, calculate T_{ih}, the total swapping cost for the pair of objects (i,h), as $T_{ih}=\sum_{1}^{n} C_{jih}$, where C_{jih} is the cost change for an objects j while swapping selected object i with non-selected object h [18].

4.2 Urban Functional Area Identification Indexes

To identify the main function of each cluster, we define the three indicator indexes, clustering density, the ratio index, and enrichment factor to judge the attribute of functional areas.

The clustering density of grid is defined as follows:

$$\rho_i = \frac{m_i}{n_i} \tag{2}$$

where m_i is the total number of POI in the i-th cluster, n_i is the number of grid in the i-th cluster

The ratio index, which is the proportion of each category of POI in one cluster, is defined as follows:

$$r_i^k = \frac{N_i^k}{N_i} \tag{3}$$

where N_i is the total number of POI in the i-th cluster, N_i^k is the number of k-th POI in the i-th cluster.

The enrichment factor (EF) of each cluster, which could contribute to the understanding of its function, is defined as follows:

$$EF_i^k = \left(N_i^k / N_i\right) / \left(N^k / N\right) \tag{4}$$

where EF_i^k represents the enrichment factor of k-th POI in the i-th cluster, N_i is the total number of POI in the i-th cluster, N_i^k is the number of k-th POI in the i-th cluster. N^k represents the total number of k-th POI, and N represents the total number of POI throughout the city.

5 Result and analysis

In this paper, we determine the number k of clusters according to experience. The k value is set from 3 to 10 to test. The results show when k=6, the clustering effect is the most reasonable. The clustering results are shown in Fig. 2.

In Table 3, we can see the grid number, POI number, and clustering density of each cluster. Although the cluster (cluster1) contains a relatively small number of grids, its clustering density reaches the maximum. There are 26 grids in this cluster, which are located in core business districts of Lanzhou City. The clustering density values of the cluster (cluster2) and the cluster (cluster3) are relatively close. These

grids are mainly distributed around the business districts. The cluster (cluster6) has a small clustering density and less POI, and it is the rural-urban fringe or the industrial area.

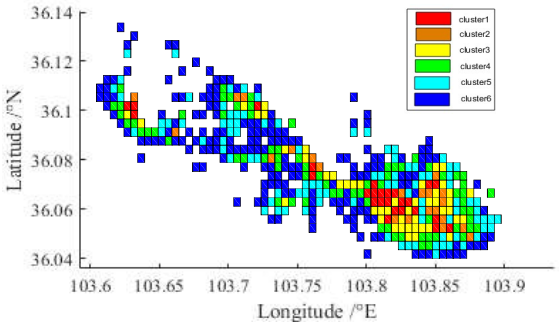


Fig. 2. Clustered pattern of urban functional areas.

Table 3. Clustering density of the cluster

Cluster	Grid number	POI number	Clustering density
1	26	30054	1155.92
2	22	11877	539.86
3	48	25627	533.90
4	73	19117	261.88
5	95	10440	109.90
6	179	2442	13.64

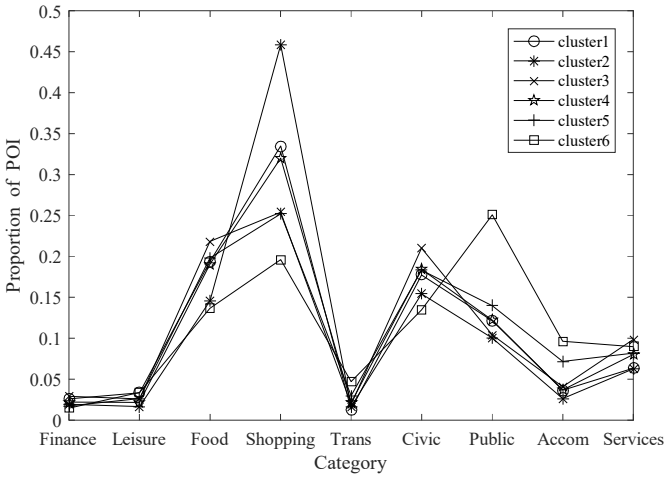


Fig. 3. The ratio index of different POI types in each cluster

The ratio index of each POI category in each cluster is shown in Fig. 3. Except for the cluster, the ratio index of the 'Shopping' POI category has the largest proportion in other clusters. If according to the ratio index to determine urban functional areas, the function of these areas can be set for shopping. However, it is clear that this conclusion is inconsistent with the actual situation. Therefore, the function of urban area cannot be determined simply in ratio index.

The ratio index of the POIs only cannot completely determine the characteristics of each cluster because it is highly possible that some POI categories have high density throughout the city. To avoid bias from the ratio index, we calculate the enrichment factor of each POI category in each cluster. The enrichment factor (EF) and normalized EF (NEF) are shown in Table 4. The function discriminant analysis can be performed. We count the number of POI categories whose NEF value is greater than 10% in each cluster. If the number of POI categories is 1, this cluster is a single functional area, otherwise it is a mixed functional area. The type of this mixed functional area is determined by the properties of the selected POI categories.

Table 4. The EF and NEF of each POI categories in each cluster

POI Category	cluster 1		cluster 2		cluster 3		cluster 4		cluster 5		cluster 6	
	EF	NEF	EF	NEF	EF	NEF	EF	NEF	EF	NEF	EF	NEF
Finance	1.08	0.1858	0.81	0.0405	1.21	0.3778	0.91	-0.2444	0.73	-0.3850	0.62	-0.3898
Leisure	1.26	0.4962	0.62	0.2641	0.98	-0.1333	0.83	-0.5111	0.99	-0.1223	1.27	-0.0307
Food	1.00	0.0479	0.75	-0.1111	1.13	0.2000	1.00	0.0556	1.02	-0.0920	0.71	-0.3401
Shopping	1.07	0.1686	1.47	0.7359	0.81	-0.5111	1.00	0.0556	0.81	-0.3042	0.63	-0.3843
Trans.	0.68	-0.5038	0.85	0.0065	0.99	-0.1111	1.13	0.4889	1.53	0.4231	2.43	0.6102
Civic & Enterprise	0.96	-0.0211	0.83	-0.0170	1.14	0.2222	1.02	0.1222	0.99	-0.1223	0.73	-0.3290
Public	1.02	0.0824	0.84	-0.0052	0.87	-0.3778	1.01	0.0889	1.17	0.0595	2.1	0.4279
Accom-modation	0.87	-0.1762	0.63	-0.2523	0.97	-0.1556	0.91	-0.2444	1.72	0.6150	2.3	0.5384
Services	0.81	-0.2797	0.8	-0.0523	1.26	0.4889	1.04	0.1889	1.04	-0.0718	1.14	-0.1025

Cluster 1 (Central Business District, CBD): This cluster is the center of the city considering the clustering density. Supplied by the data from Fig.2 and Table 2, this cluster shows high NF and NEF values for leisure (1.26, 0.4962), finance (1.08, 0.1858), and shopping (1.07, 0.1686). In addition, there are a great number of food and public organizations within this cluster. There are 26 grids in this cluster, most of which are distributed in Chengguan District. The corresponding locations are Xiguan Commercial District, Dongfanghong Plaza Commercial District, Nanguan Commercial District, Wanda Plaza Commercial District, and Railway Station Commercial District. A small part is distributed in Qilihe District, Anning District and Xigu District.

Cluster 2 (Shopping centers): We call cluster 2 the shopping center areas because the NEF values for shopping is 0.7359, which is the largest value and is much higher than that in the other POI categories. In Fig. 2, we can see that the shopping center areas are located in the center of the city and densely populated areas. There are a total of 22 grids corresponding to this area, which are mainly distributed outside the major business districts, and a few are distributed in the center or surrounding areas of different administrative areas.

Cluster 3 (Peripheral business district, PBD): This cluster lies at the peripheral of the CBD. In this cluster, there are four POI categories for NEF greater than 10%. The four POI categories include services, finance, Civic & Enterprise, and food. There are 48 grids this cluster, which are distributed around the CBD and shopping centers.

Cluster 4 (Traditional residential area): This cluster lies at the peripheral of the PBD and has a smaller POI density. The variance of EF values is relatively small which indicate the cluster has a balanced POI configuration. There are four classes with high NEF values, which include Trans., services, and Civic & Enterprise..

Cluster 5 (Developing residential area): This cluster has a low POI density. The NEF values of two POI categories (Trans and Accommodation) are larger than that of the other POI categories. This cluster mainly distributed in the periphery of each administrative district, which is close to the central area of the city, with convenient transportation and more land resources. They are typical developing residential areas.

Cluster 6 (urban-rural fringe): This cluster has a low POI density. Most of these areas are distributed urban-rural fringe. This region cluster shows high NEF values for trans. (0.6102), accommodation (0.5384) and public (0.4279). These areas are far from the commercial areas of the city, but have relatively complete infrastructure. It is a typical urban connection zone. This area is the most complex and fastest-changing area in the city. It is also an active zone where urban and rural functions are interdoped and interpenetrated.

6 Conclusion

It is practical significance to identify urban functional areas and spatial distribution characteristics. The POI data contains rich information on urban spatial structure and is an important data source for exploring urban spatial layout. Based on the POI data of Lanzhou City, this paper proposes an urban functional area recognition algorithm based on PAM cluster analysis and enrichment factor. This method can effectively identify the urban functional area of Lanzhou. In the future research, the other adaptive factors, such as traffic data, should be considered. In addition, in the process of urban functional area analysis, the scale should be considered in the grid division of urban functional areas. The size of the cell grids may have a slight impact on the conclusions of the study.

Acknowledgment

This research was supported by the National Natural Science Foundation of China (Grant No. 71761031).

References

1. Yang, S.: A study on population distributing and function area in Shanghai. Capital University of Economics, 2007.
2. Dou, Z.: Research on spatial clustering algorithm for urban functional area division. Sichuan Normal University, 2010.
3. Li, Y.: Research on urban functional semantic partition based on remote sensing and POI data. University of Chinese Academy of Sciences, 2018.
4. Cui, C., Wang J., Wu, Z., Ni, J., Qian, T.: The socio-spatial distribution of leisure venues：A case study of karaoke bars in Nanjing. International Journal of Geo-Information 1(5), 2-17 (2016).
5. Xue, B., Xiao, X., LI, J., Xie, X., Lu, C., Ren, W.: POI-based spatial correlation of the residences POI-based spatial correlation of the residences and retail industry in Shenyang City. Scientia Geographica Sinica (39), 442-449 (2019).
6. Jia, X., Lei, J. Wu, W .: Urban recreation space pattern analysis based on POI data: A case study of Urumqi. Arid Land Geography.1-15 (2019).
7. Zhu, Y.: Analysis on the spatial pattern of catering facilities in Nanjing based on POI data. Economic Research Guide (15), 152-156 (2019).
8. Wang, W., Bai, Y., Lu, J.: The classification and coupling of commercial space vitality around subway station based on point of interest (POI) data: A case study of Tianjin. City (5), 14-22(2019).
9. Zhang, M., Zhang, E., Shan, Z.: Research on the identification of multiple types of commercial center and spatial patterns in Wuhan based on POI data. South Architecture (2), 55-61 (2019).
10. Chen, Y., Liu, L., Liang, Y.: Retail center recognition and spatial aggregating feature analysis of retail formats in Guangzhou based on POI data. Geographical Research (35), 703-716 (2016).
11. Lin, Q., Sun, F., Wang, X., Liao, C., Zhang, W.: Hierarchical system of Beijing commercial center judged from POI data. Journal of Beijing Normal University.1-10 (2019).
12. Hao, F., Wang, S., Feng, Z.,et al.: Spatial pattern and its industrial distribution of commercial space in Changchun based on POI data. Geographical Research (37), 366-378 (2018).
13. Chi, J., Jiao, L., Dong, T., Gu, Y., Ma, Y.: Quantitative identification and visualization of urban functional area based on POI data. Journal of Geomatics (41), 68-73 (2016).
14. Li, J., Liang, Y., Wang, X.: Spatial cluster analysis of service industries in Zhengdong New District based on POI data. Geographical Research (37), 145-15 7(2018).
15. Zhang, X., Du, S., Wang, Q.: Hierarchical semantic cognition for urban functional zones with VHR satellite images and POI data. Journal of Photogrammetry and Remote Sensing (132) ,170-184 (2017).
16. Zhai, W., Bai, X., Shi, Y., Han, Y., Peng, Z., Gu, C.: Beyond Word2vec: An approach for urban functional region extraction and identification by combining Place2vec and POIs.Computers, Environment and Urban Systems (74), 1-12(2019).
17. Perner, P. Lecture Notes in Artificial Intelligence, Vol. 2558. Springer Verlag, Berlin Heidelberg New York (2002), ISBN: 3-540-00317-7.
18. Seman, A., Bakar, Z. A. A. Mohd. Et al. A medoid-based method for clustering categorical data, Journal of Artificial Intelligence(6), 257-265 (2013).

Measuring the effectiveness of code review comments in GitHub repositories: A machine learning approach

Shadikur Rahman, Umme Ayman Koana, Hasibul Karim Shanto, Mahmuda Akter, and Chitra Roy

{Daffodil International University, Bangladesh University of Engineering and Technology, Ahsanullah University of Science and Technology}, Dhaka, Bangladesh
Khulna University of Engineering & Technology, khulna, Bangladesh
shadikur35-988@diu.edu.bd, {koana2k12, hasibsourov36}@gmail.com, mahmuda_akter03@yahoo.com, Chtrroy13@gmail.com

Abstract. This paper illustrates an empirical study of working efficiency of machine learning techniques in classifying code review text by semantic meaning. The code review comments from the source control repository in GitHub were extracted for development activity from the existing year for three open-source projects. Apart from that, Programmer's need to be aware of their code and point out their errors. In that case, it is a must to classify the sentiment polarity of the code review comments for avoiding an error. We manually labeled 13557 code review comments generated by three open source projects in GitHub during the existing year. In order to recognize the sentiment polarity (or sentiment orientation) of code reviews, we use seven machine learning algorithms and compare those results to find the better ones. Among those Linear Support Vector Classifier(SVC) classifier technique achieves higher accuracy than others. This study will help the programmer's to make any solution based on code reviews by avoiding misconceptions.

Keywords: Sentiment Analysis · Machine learning · Semantic Orientation · Code review · Text Mining

1 Introduction

The code review comments are the key significative of the substance of any coding bug issue or error. By reading any review comments, programmers can understand any coding errors. In most cases, developers do not read the whole code as they think the review comment is the essence of any errors. Sometimes programmers are misguided by reading the review comment on any coding issues. In [Bosu et al.(2015)Bosu, Greiler, and Bird], it is investigated that how code review comments useful the coder logic and influencing them towards error solving. In that case, it is a must to identify the sentiment polarity of the review comments for avoiding misconception.

Depending on the context of the review comments, the semantic orientation would be different. Moreover, any specific comments sometimes presented differently in different review comments. So it is essential to find the semantic orientation of any code review comment. There are missing enough studies to find out the semantic orientation of a review comment based on the context of coding bug. Open source software measures peer code review impact in [Rigby et al.(2012)Rigby, Cleary, Painchaud, Storey, and German], the method of analyzing code written by the different developers on the project to decide whether it is of sufficient quality to be integrated into the project codebase.

In this study, we present the effectiveness of machine learning and deep learning techniques to find the semantic orientation of any review comments. We use Naïve Bayes [Rish et al.(2001)], Multinomial Naïve Bayes (MNB) [Rennie et al.(2003)Rennie, Shih, Teevan, and Karger], Bernoulli Naïve Bayes [McCallum et al.(1998)McCallum, Nigam, et al.], logistic regression [Ho et al.(1994)Ho, Hull, and Srihari], Stochastic Gradient Descent (SGD) [Bottou(2010)], Linear Support Vector Classifier(SVC) [Gunn et al.(1998)], and Nu support vector classifier [Schölkopf et al.(2000)Schölkopf, Smola, Williamson, and Bartlett], and Word2vec method [Goldberg and Levy(2014)] to find out the most appropriate semantic orientation. The result of this study will help programmers.

This paper is structured as follows: Related work is described at 2 that followed by Research Methodology and Result & Discussion at 3 and 4 respectively. The final section 5 summarizes our contribution and furnishes the conclusion.

2 Related Work

The sentiment analysis of code review comments from GitHub projects repository consists of a big stages: the sentiment analysis of all the code review comments.It is shown [Pletea et al.(2014)Pletea, Vasilescu, and Serebrenik] that the sentiment analysis of the detection of comments related to the security topic and the sentiment analysis of all the comments from GitHub projects. Many works have been done on product reviews using opinion mining and sentiment analysis. In [Rahman et al.(2019)Rahman, Hossain, Islam, Chowdhury, Rafiq, and Badruzzaman] that most of the sentiment analysis was done on news headlines data. Other than that news data, social media data and web blogs data are also used. In [Pang et al.(2002)Pang, Lee, and Vaithyanathan], Naïve Bayes classification, maximum entropy classification, and support vector machines these tree machine learning algorithm were used to perform sentiment analysis on movie review. A feature-based opinion summarizing technique is presented in [Hu and Liu(2004)] of product reviews using data mining and natural language processing. A system Opinion Observer [Liu et al.(2005)Liu, Hu, and Cheng] is implemented using proposed holistic lexicon-based approach [Ding et al.(2008)Ding, Liu, and Yu]. A lexicon-based approach is used in [Im et al.(2013)Im, San, On, Alfred, and Anthony] to identify the positive or negative polarity of the financial news. A large-scale sentiment analysis system is presented in this paper [Godbole et al.(2007)Godbole, Srinivasaiah, and Skiena] to indicate positive and negative

opining of news and blogs by assigning score. In [Maynard et al.(2012)Maynard, Bontcheva, and Rout] the author performed a sentiment analysis by indicating positive, negative or neutral sentiments. Sentiment Analysis tools based on SentiCR In [Ahmed et al.(2017)Ahmed, Bosu, Iqbal, and Rahimi] of social media text or product reviews. Opinion Lexicon-based algorithm and Naïve Bayes algorithm is used in [Shuhidan et al.(2018)Shuhidan, Hamidi, Kazemian, Shuhidan, and Ismail] for sentiment analysis of financial news headlines of Malaysia.

In recent, deep learning is also used for sentiment analysis. Analyzing emotion hidden of the microblog messages in [Xue et al.(2014)Xue, Fu, and Shaobin] Sentiment Dictionary using Word2vec tool based on our Semantic Orientation. In [Severyn and Moschitti(2015)] uses deep learning approach for predicting polarities of tweets at both message level and phrase level. Micro blogging and movie reviews datasets are used in [Araque et al.(2017)Araque, Corcuera-Platas, Sanchez-Rada, and Iglesias] to measure the performance of sentiment analysis using deep learning. In [Tang et al.(2015)Tang, Qin, and Liu], authors provide an overview of deep learning approaches for sentiment analysis and also suggest some mitigation to address the challenges.

3 Research Methodology

In this section, The entire process of our research activities has been described. First of all, we have chosen our dataset[1]. For the dataset, we have selected code review comments to perform our research process. For preparing so, we have a lot of covers to process e.g. step by step preprocessing, chunking, N-gram, training model with the help of machine learning and deep learning methods. We convey a retrieve responsibility to compare two topic representations: (1) Sentiment analysis (2) Word scoring and (3) Measuring comments. Figure 1 shows an overview of our research experiment.

3.1 Training Dataset Generation

We utilized according to the three-step approach to build a labeled sentiment dataset from code review comments. We focused only on the structure code review comments. Table 1 summarizes the statistics about these three projects data sets.

3.1.1 GitHub Api: We used GitHub api[2] to mine the code review comments of our three popular projects.We used the repos-url to fetch all repositories of our projects. We had worked on several languages including HTML, CSS, C++, Java, Python, and others. Our projects have performed at least 5,000 code reviews.

[1] https://github.com/sadirahman/Code-review-sentiment-analysis/tree/master/code-reviews

[2] https://api.github.com/

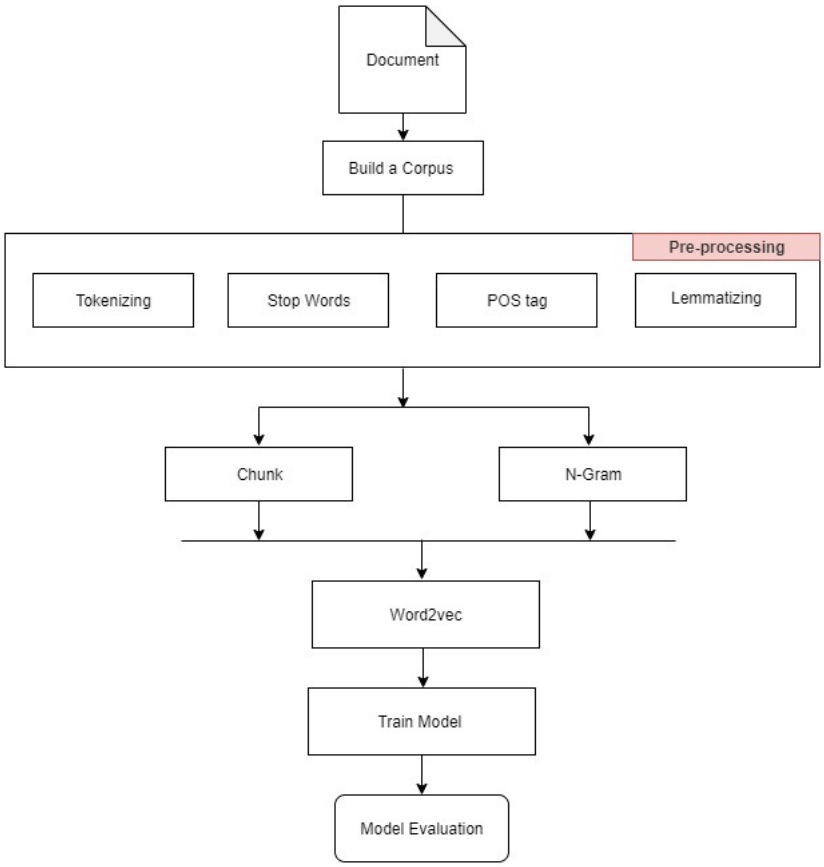


Fig. 1. Research Methodology

Table 1. Statistics of Data sets

Project Name	Time length	Comments	Ratio
Alpha	Jan, 2019 to Nov, 2019	4867	35.90%
Beta	Jan, 2019 to Nov, 2019	5655	41.71%
Gamma	Jan, 2019 to Nov, 2019	3035	22.38%
	Total	13557	100%

3.2 Data Categorizing

After collecting data sets, we categorize the comments based on structure reviews. For this paper, we mainly consider only 4 categories. For doing this, we categorize the comments using the meta data of each project's URL. Table 2 shows the categories of the reviews of we identified. The distribution of the number of reviews items for each project is shown in Table 3.

Table 2. Categories of code review comments

Labels	Comments
Efficient	Please reformat in accordance to previous class
Not-Efficient	Sure sounds good
Some-How-Efficient	I think this information is obsolete now
System-Generated	Change has been successfully merged by Sadi Rahman

Table 3. Distribution of review comments based on labels

Labels	Alpha	Beta	Gamma
Efficient	44%	31%	29%
Some-How-Efficient	30%	19%	23%
Not-Efficient	11%	14%	13%
System-Generated	15%	36%	35%

3.3 Data Preprossessing

For analysis our data set, we preprocess all the review comments. In most of the cases, we realize that text data is not perfectly cleaned. For cleaning the text data, text pre-processing is needed. For doing the pre-processing, we need to follow several steps like tokenization, stop words removing, POS tag, lemmatizing and removing punctuation. Figure 2 shows an overview of our data prepossessing process.

3.3.1 Text Tokenization: Tokenization is the method of splitting the provided text into smaller portions called token. Words, numbers, punctuation marks, and others can be recognized as the token.

Fig. 2. Data Preprossessing process

3.3.2 Stop Words Removing: Stop words are the usual common words in a language like “about”, “an”, “the”, “is”, “into”. These words do not give important meaning and are normally removed from text documents of our dataset.

3.3.3 Lemmatizing: We used Lemmatizing process of decreasing words to their word Lemma, base or root form, for example, roads–road, loved–love.

3.3.4 Removing punctuation: Remove punctuation is needed if they are not related to the text corpus. Normally, regular expressions are used to remove set of punctuation symbols.

3.3.5 POS tag: We used NLTK word tokenizer to parse each text into a list of words.After that Text Tokenization,then we used Pos tagging in NLP using NLTK. The part of speech(POS tag) explains corpus how a word is used in a sentence.POS tag is separated into subclasses. POS Tagging solely means labeling words with their appropriate Part-Of-Speech.There are eight main parts of speech - nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions and interjections.Parts of Speech (POS) tag from Penn Tree bank annotation [Taylor et al.(2003)Taylor, Marcus, and Santorini] is shown in Table 4.

Table 4. POS tagging Penn Tree bank annotation

Types of POS	Initial	Examples
Noun	N	Daniel, table, happiness, hope
Verb	V	go, run, live, like, are
Adjective	ADJ	big, happy, green, young
Adverb	ADV	quietly, very, always, never
Preposition	P	at, on, in, from
Conjunction	CON	but, because, so, yet
Pronoun	PRO	you, we, they, he
Interjection	INT	Ouch!, Wow!, Great!, Help!

3.4 Chunking Process

After the POS tag, we used the chunking process of extracting phrases from unstructured corpus text.Chunking works on top of POS tagging, it uses pos-tags as input and provides chunks as output.We search for chunks corresponding to an individual some POS tag phrase. The review comments contains many parts of speech which are irrelevant to detect semantic orientation in our case. We consider only Adjectives (JJ), Verb (VB), Adverb (ADV) and Noun (NN) Parts of Speech (POS) tag from Penn Tree bank annotation.For example,code review comment is "I think this information is obsolete now".Figure 3 shows an overview of our chunking process

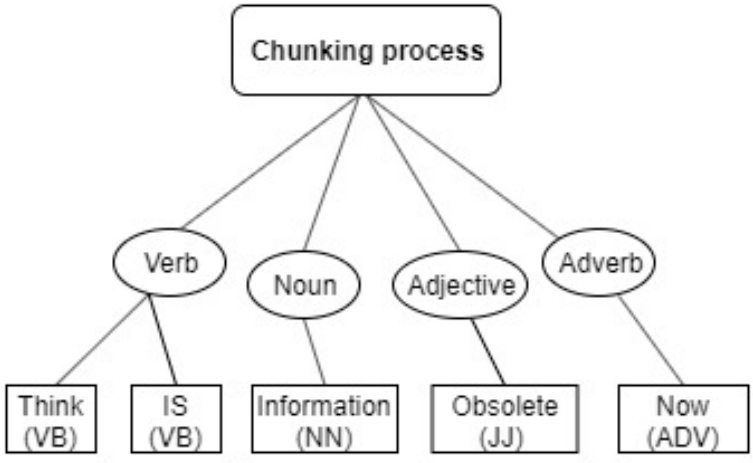


Fig. 3. Overview of Chunking Process

3.5 N-Gram Process

An N-gram is a sequence of N words, which computes p(w—h), the probability of a word w* [Brown et al.(1992)Brown, Desouza, Mercer, Pietra, and Lai]. We have used N-gram model using the same training text and held-out data as we used for the word-based Natural language process model we discussed above in our research. The purpose of using this is to maintain the sequence of candidate labels in our training data sets. For example, if we consider the text- "Please enter your comment". For this text, the 2-gram and 3-gram words is presented in Table 5.

Table 5. Process of an N-gram words

2-gram(bigram)	3-gram(trigram)
"Please enter"	"Please enter your"
"Enter your"	"Enter Your comment"
"Your comment"	

3.6 Word2vec Process

After the preprocessing, we used the word2vec method in our research work. Word2vec gives direct access to vector representations of training corpus words. Word2vec is to classify the vectors of similar words together in vector space. it recognizes similarities mathematically. Word2vec produces vectors that are distributed numerical representations of word features, features such as the context of individual words. train Word2Vec Model based on code review corpus and gensim word2vec module. We used word2vec for the most similar word-finding parameter "topn = 10".For example, the word "error" is a vector representation Table 6.

Table 6. Process of Word2vec

Similar Words(Error)	Numerical Representations
Holes	0.93
Necessity	0.91
Drought	0.91
Storm	0.88
Blade	0.88
Flies	0.88
Washing	0.87
Cluster	0.87
Distresses	0.87
Lazarus	0.87

3.7 Training Model

In this section, for training our model, we have used seven machine learning algorithms. Used algorithms are:Naïve Bayes [Rish et al.(2001)], Multinomial Naïve Bayes (MNB) [Rennie et al.(2003)Rennie, Shih, Teevan, and Karger], Bernoulli Naïve Bayes [McCallum et al.(1998)McCallum, Nigam, et al.], Logistic Regression [Ho et al.(1994)Ho, Hull, and Srihari], Stochastic Gradient Descent (SGD) [Bottou(2010)], Linear Support Vector Classifier(SVC) [Gunn et al.(1998)], and Nu support vector classifier [Schölkopf et al.(2000)Schölkopf, Smola, Williamson, and Bartlett]. The data set for training model is available at [3].

We have trained our code review document corpus data sets with our chosen train models. Figure 4 shows the training process of our models

[3] https://github.com/sadirahman/Code-review-sentiment-analysis

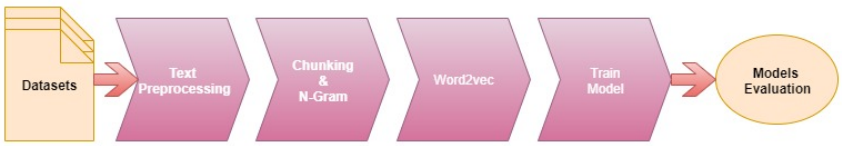


Fig. 4. Training Process of models

4 Result & Discussion

In this section, we explain the classification results, the evaluation contexts and the polarity estimation of our research.

4.1 Classification Results

In order to classify the code review comments as Efficient or Not-Efficient or Some-How-Efficient or System-Generated, we used seven different classification algorithms. Table 7 shows the accuracy of different classifiers we used. It is

Table 7. Accuracy of different classifiers

Algorithm	Accuracy (%)
Naïve Bayes	81.32
Multi nomial Naïve Bayes (MNB)	82.76
Bernoulli Naïve Bayes	80.11
Logistic Regression	82.50
Stochastic Gradient Descent (SGD)	78.55
Linear Support Vector Classifier(SVC)	83.09
Nu support vector classifier	79.75

terrible to see some of the widely used algorithms fail to achieve satisfactory performance for this case. Most notably, the Naïve Bayesian, Linear Support Vector Classifier(SVC), Multi nomial Naïve Bayes (MNB), Bernoulli Naïve Bayes and Logistic Regression classifiers achieve an average accuracy of about 81%. However, other variants of Stochastic Gradient Descent (SGD) and Nu support vector classifier gave accuracy of more than 75% but less than 80% in some cases. The Linear Support Vector Classifier(SVC) classifier gives the best accuracy.

4.2 Corpus Validation

In our corpus datasets that we used to train the models, contains 13557 comments (Efficient-3326, Not-Efficient-419, Some-How-Efficient-975,and System-Generated-8837 lines comments). We train our models considering 75% as training dataset of our code review corpus and test the models with remaining 25%

test dataset of corpus. Test dataset is used to provide an unbiased evaluation of a final model to fit with the training dataset. Section 4.1 shows the details the validation results.

4.3 Sentiment Estimation

In this section, we explain the Sentiment results, the evaluation contexts and the polarity estimation of our research.

4.3.1 Sentiment Consideration

We present the words polarity in Table 8. Our train model gives most informative features words in trained data sets. In the first instance, model gives informative words"classifier.show-most-informative-features()" base on our sentiment ratio score.

Table 8. Sentiment Ratio

Words	Sentiment	Sentiment Ratio
Need	Efficient	92.8
Please	Efficient	98.5
Merged	System-Generated	65.7
Ok	Not-Efficient	78.5
Thanks	Not-Efficient	85.6
Think	Some-How-Efficient	82.7

4.4 Precision, Recall and F-measure for Code Reviews Data sets

As we found highest accuracy with Linear Support Vector Classifier(SVC), so we trained our model by Linear Support Vector Classifier(SVC). This time, we run random shuffle in our trained date sets (Efficient,Not-Efficient,Some-How-Efficient and System-Generated). This classifier gives the accuracy measuring the trained data set. We collect the reference values and observed values for each label (Efficient,Not-Efficient,Some-How-Efficient and System-Generated), then use those sets to calculate the precision, recall, and F-measure of the Linear Support Vector Classifier(SVC). Shows the resultant confusion matrix is a table 9 that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

4.5 Sentiment Polarity Considering Procedure

In this section, train model gives sentiment against our code review comments. Instance model gives comments sentiment based on our training models.We will multiply the answer by 100% so it's in a percentage and we will even denote this accuracy percent. Our models give comments sentiment the confidential score between 0 and 1.0 it gives the result as sentiment. Table 10 shows some of the confidential score of code review comments.

Table 9. Confusion Matrix of code Reviews Data sets

Sentiment	Precision	Recall	F-measure
Efficient	0.67	0.81	0.73
Not-Efficient	0.64	0.58	0.60
Some-How-Efficient	0.62	0.68	0.64
System-Generated	0.74	0.78	0.75

Table 10. Sentiment result of comments

Review comments	Sentiment	Confidential score
Please maintain same coding rules applied in this project.	Efficient	0.7
Sure I will change it.	Not-Efficient	0.6
Should there be a new line? please check.	Some-How-Efficient	0.7
Change has been successfully merged by Rifat Hasan	System-Generated	0.8

5 Conclusion

The review comments are the influential element of any bug issue of code. In this paper, we proposed a technique to identify the code review comments from GitHub repositories projects in terms of semantic orientation using a machine learning approach. The objective is to find out the context based tagging of review comments to avoid prejudicing the coder. Our work will mostly help the programmers to make decision based on review comments by avoiding misconception . Moreover, this technique can also be implemented in a tool for identifying GitHub or Gerrit project repositories.

In the future, current work can be extended to analyze the review comments and developer reviews together to find out the most authenticate comments on GitHub.

References

Bosu et al.(2015)Bosu, Greiler, and Bird. Amiangshu Bosu, Michaela Greiler, and Christian Bird. Characteristics of useful code reviews: An empirical study at microsoft. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 146–156. IEEE, 2015.

Rigby et al.(2012)Rigby, Cleary, Painchaud, Storey, and German. Peter Rigby, Brendan Cleary, Frederic Painchaud, Margaret-Anne Storey, and Daniel German. Contemporary peer review in action: Lessons from open source development. *IEEE software*, 29(6):56–61, 2012.

Rish et al.(2001). Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

Rennie et al.(2003)Rennie, Shih, Teevan, and Karger. Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

McCallum et al.(1998)McCallum, Nigam, et al.. Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

Ho et al.(1994)Ho, Hull, and Srihari. Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):66–75, 1994.

Bottou(2010). Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Gunn et al.(1998). Steve R Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

Schölkopf et al.(2000)Schölkopf, Smola, Williamson, and Bartlett. Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

Goldberg and Levy(2014). Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Pletea et al.(2014)Pletea, Vasilescu, and Serebrenik. Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351. ACM, 2014.

Rahman et al.(2019)Rahman, Hossain, Islam, Chowdhury, Rafiq, and Badruzzaman. Shadikur Rahman, Syeda Sumbul Hossain, Saiful Islam, Mazharul Islam Chowdhury, Fatama Binta Rafiq, and Khalid Been Md Badruzzaman. Context-based news headlines analysis using machine learning approach. In *International Conference on Computational Collective Intelligence*, pages 167–178. Springer, 2019.

Pang et al.(2002)Pang, Lee, and Vaithyanathan. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Hu and Liu(2004). Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

Liu et al.(2005)Liu, Hu, and Cheng. Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

Ding et al.(2008)Ding, Liu, and Yu. Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*, pages 231–240. ACM, 2008.

Im et al.(2013)Im, San, On, Alfred, and Anthony. Tan Li Im, Phang Wai San, Chin Kim On, Rayner Alfred, and Patricia Anthony. Analysing market sentiment in financial news using lexical approach. In *2013 IEEE Conference on Open Systems (ICOS)*, pages 145–149. IEEE, 2013.

Godbole et al.(2007)Godbole, Srinivasaiah, and Skiena. Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *Icwsm*, 7(21):219–222, 2007.

Maynard et al.(2012)Maynard, Bontcheva, and Rout. Diana Maynard, Kalina Bontcheva, and Dominic Rout. Challenges in developing opinion mining tools for social media. *Proceedings of the@ NLP can u tag# usergeneratedcontent*, pages 15–22, 2012.

Ahmed et al.(2017)Ahmed, Bosu, Iqbal, and Rahimi. Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. Senticr: a customized sentiment analysis tool for code review interactions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 106–111. IEEE Press, 2017.

Shuhidan et al.(2018)Shuhidan, Hamidi, Kazemian, Shuhidan, and Ismail. Shuhaida Mohamed Shuhidan, Saidatul Rahah Hamidi, Soheil Kazemian, Shamila Mohamed Shuhidan, and Maizatul Akmar Ismail. Sentiment analysis for financial news headlines using machine learning algorithm. In *International Conference on Kansei Engineering & Emotion Research*, pages 64–72. Springer, 2018.

Xue et al.(2014)Xue, Fu, and Shaobin. Bai Xue, Chen Fu, and Zhan Shaobin. A study on sentiment computing and classification of sina weibo with word2vec. In *2014 IEEE International Congress on Big Data*, pages 358–363. IEEE, 2014.

Severyn and Moschitti(2015). Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM, 2015.

Araque et al.(2017)Araque, Corcuera-Platas, Sanchez-Rada, and Iglesias. Oscar Araque, Ignacio Corcuera-Platas, J Fernando Sanchez-Rada, and Carlos A Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236–246, 2017.

Tang et al.(2015)Tang, Qin, and Liu. Duyu Tang, Bing Qin, and Ting Liu. Deep learning for sentiment analysis: successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6):292–303, 2015.

Taylor et al.(2003)Taylor, Marcus, and Santorini. Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer, 2003.

Brown et al.(1992)Brown, Desouza, Mercer, Pietra, and Lai. Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

The complex role of location and time in prediction models

Mahdi Hashemi [0000-0003-0212-0228]

George Mason University, Fairfax VA 22030, USA
mhashem2@gmu.edu

Abstract. This study poses the question, how the recorded location and time for training samples should contribute in training and testing a machine? Historically, location and time have been either (a) dismissed altogether, (b) considered as the only input features, or (c) considered as additional input features along with other non-spatial and non-temporal features, when training or testing machine learning techniques. These three strategies have resulted in varying generalization accuracies in different experiments in the literature, yet none of them has undisputedly dominated the others. This study argues that existing approaches fail to properly capture the way in which spatial-temporal phenomena behave in reality. Consequently, new or modified theoretical machine learning models need to be developed to properly exploit the prior knowledge of how spatial-temporal data behave. This paper provides an exhaustive review of literature with regard to this problem, after describing the problem in more detail, followed by proposing a new methodology to embed the external knowledge of spatial autocorrelation into weighted machine learning models. While we theoretically show that the proposed approach captures the spatial autocorrelation more precisely, this is also confirmed by its higher accuracy in experiments with two datasets.

Keywords: Inductive Learning, Analytical Learning, Machine Learning, Spatial Data, Temporal Data, Autocorrelation.

1 Introduction

Spatial autocorrelation states that: “Data from locations near one another in space are more likely to be similar than data from locations remote from one another” [1,2,3]. Spatial autocorrelation is the result of first- and second-order effects in spatial processes [1]. First-order effects refer to environmental effects and second-order effects refer to interactions between samples. Temporal data show a similar behavior [4,5], referred to as temporal autocorrelation. In addition to this linear autocorrelation, temporal data might also indicate a cyclic change [6,7,8,9,10,11,12], referred to as cyclic temporal autocorrelation. Because of spatial and temporal autocorrelations, spatial-temporal data are not truly random. In other words, phenomena do not vary randomly through space and time.

Spatial autocorrelation and temporal autocorrelation are the backbone of spatial and temporal data analytics. For example, if the temperature at location A is 30°C, the temperature at location B, 1 m away from A, is also 30°C, the temperature at location C, 100 m away from A would be very close to 30°C, and the temperature at location D, 2 km away from A, is more uncertain and can be different (less or more) than 30°C. More examples can be found in the related literature [13,14,15]. This example can also be used to explain the temporal autocorrelation; the longer the time difference, the more uncertain we are about the temperature at location A, which was measured only at a snapshot. Temperature also has a strong cyclic temporal autocorrelation, i.e. temperature rises from winter to spring and keeps rising until summer, then falls from summer to autumn and keeps falling until winter. This cycle repeats itself every year. More examples of cyclic temporal autocorrelation can be found in the literature [6,7,8,9,10], although it is referred to by different names. This cyclic behavior makes temporal autocorrelation more challenging to model than spatial autocorrelation. Other examples of spatial-temporal data are elevation, air or water pollution, soil type, weather, population, landuse, and landslide.

This work is an attempt to not only review existing methodologies for considering location and time in prediction models, but also to explore more effective approaches for doing so. Section 2 reviews select literature regarding the prediction of spatial-temporal phenomena and highlights their shortcomings. Section 3 explains our proposed methodology to best embed spatial-temporal autocorrelation as external knowledge into machine learning models. Section 4 compares the proposed method's results with those of existing approaches. Section 5 concludes this study, while providing future research directions.

2 Related Work

Franklin [16], in her review paper, introduced the spatial dependence/autocorrelation as a source of information which has yet to be exploited in vegetation prediction models. O'Sullivan and Unwin [1] raised the concern with respect to blindly applying machine learning techniques to spatial data. In their book on geographic information analysis, they elaborated on their concern that special characteristics of spatial data are ignored in regression and classification models. Shekhar et al. [17] showed that spatial autocorrelation limits the usefulness of conventional classification and regression techniques for extracting spatial patterns. Santibanez et al. [13,14] also raised this concern by stating that "machine learning algorithms are in general, not designed to deal with spatially autocorrelated data." The assumption of independent and identically distributed random variables is not valid for spatial data because of spatial autocorrelation. Spatial autocorrelation causes the prediction residuals to exhibit clustering over the geographic space [13,14,15,18,19]. In an attempt to address this issue, some researchers [15] suggested a spatial version of least squares (LS) model which computes the weight vector as $(X^TCX)^{-1}X^TCy$. In this equation, C is defined as an indicator of spatial neighborhood among observations, X as the feature matrix, and y as the target variable. Other extensions of the LS model, attempting to incorporate the spatial neighborhood, have

also been proposed in the literature [20] and have been able to improve the prediction accuracy. However, the concern is not fully addressed in these works since they apply spatial neighborhood rather than spatial autocorrelation, do not consider the time, and do not go any further than the LS model. On the other hand, spatial interpolation methods such as kriging or k-nearest neighbors (kNN) are insufficient to address this concern because they operate in the Euclidean space rather than the feature space [1,17] and they ignore non-spatial features. Ignoring non-spatial features make these approaches unreliable when spatial autocorrelation is weak or useless when the test sample is spatially far from training samples.

On the other hand, some researchers showed the reversibility (cyclic behavior) of landuse changes [6,7,8,9] and earthquakes [12,11] in time. Mertens and Lambin [6] showed that landuse predictions are more reliable in long term when more historic training samples are available. However, developing machine learning techniques that capture the idiosyncratic behaviors of spatial-temporal phenomena remains a challenge.

Machine learning techniques learn to distinguish among patterns based on features. These patterns are recognized by measuring the similarity among training samples' features. Spatial-temporal data record the location and time of each observation along with other features. Literature on machine learning for spatial-temporal data applies two general strategies in deploying location and time: (a) considering them as input features, and (b) ignoring them altogether. This section reviews select literature applying the first strategy and then provides a list of select literature applying the second strategy.

Li et al. [21] showed that combining machine learning techniques such as random forest (RF), regression tree (RT), or support vector machines (SVM) with spatial interpolation methods such as kriging or inverse distance squared (IDS) improves the accuracy of predicting seabed mud content in the southwest Australian margin. In these combined methods, the machine learning technique is first applied to the features, then the spatial interpolation method is applied to the prediction residuals, and finally the interpolated residuals are added to the predicted values. The input features include bathymetry, distance-to-coast, seabed-slope, latitude, longitude, as well as their second and third powers, multiplication of latitude and longitude, multiplication of latitude to the second power of longitude, and multiplication of longitude to the second power of latitude. Their results showed that RF-OK (random forest combined with ordinary kriging), RF-IDS (random forest combined with inverse distance squared), RF, and RT-OK (regression tree combined with ordinary kriging) are the most accurate methods, respectively. Combination of SVM (with a linear or Gaussian kernel) with ordinary kriging (OK) or IDS considerably improved its prediction accuracy, although it remained less accurate than OK or IDS. RF [22] was more accurate than RT, and RT was more accurate than SVM.

Kanevski et al. [18] applied a similar approach in combining machine learning techniques with geostatistical models with the difference that the spatial coordinates of observations were the only input features to the machine. They showed that nonlinear regression models including multilayer perceptron (MLP) and support vector regression (SVR) [23] with Gaussian kernel, trained with spatial coordinates as input features, could effectively capture the nonlinear global spatial trend of the target variable. However, the spatially autocorrelated prediction residuals are a manifestation of spatial

autocorrelation which was not properly captured by their model. They applied sequential Gaussian simulation as a remedy, in combination with the prediction model, to increase the prediction accuracy. In an experiment to predict the radioactive soil contamination, they reported a better generalization accuracy for the combined approach than either the machine learning technique or the geostatistical model when applied alone. In another effort to predict the radioactive soil contamination, Kanevski et al. [24] used spatial coordinates as the only input features to a kNN and a general regression neural network (GRNN). GRNN is a non-parametric regression model based on Parzen windows [25]. They considered two versions of GRNN, one isotropic where the kernel bandwidth is the same in all directions and another anisotropic where the kernel has different bandwidths in different directions. To consider different bandwidths in different directions in a kernel, a matrix is used as the bandwidth instead of a constant value. They considered two directions in their anisotropic GRNN model and optimized those directions and their bandwidths using leave-one-out cross-validation. KNN, isotropic GRNN, and anisotropic GRNN resulted in an RMSE of 22.1, 12.4, and 11.9, respectively. This result indicates that (a) assigning different weights to neighbors based on Parzen windows, which is the implemented strategy in their GRNN, improves the accuracy in comparison with assigning equal weights to all neighbors, which is the implemented strategy in their kNN, and (b) considering different bandwidths in different directions for the kernel, in anisotropic GRNN, improves the accuracy in comparison with a single bandwidth, in isotropic GRNN.

Gilardi and Bengio [19,26] compared the generalization accuracy of four regression techniques in predicting the rainfall based on spatial coordinates of observations. Their results, in line with Kanevski et al. [18] results, confirmed that global regression models, such as MLP and SVR [23] with a Gaussian kernel, trained with spatial coordinates as input features capture the nonlinear global spatial trend in the target variable but leave the local spatial autocorrelation untreated. Consequently, local regression techniques, such as mixture of experts (ME) [27] and local SVR (which is the standard SVR trained only by training samples near the test sample in the feature space), achieved slightly better generalization accuracies because they were able to partly capture the local spatial autocorrelation. They reported an RMSE of 63.4, 59, 57.1, and 53.2 for SVR, MLP, local SVR, and ME, respectively.

Table 1 provides a list of select scientific articles that ignore location and time when applying prediction models to spatial-temporal phenomena. Table 1 reports the overall accuracy for classification cases and R^2 for regression cases if they are either reported by the authors or possible to calculate from the information in their paper. Otherwise, alternative measures, such as area under the curve (AUC), precision, and Kappa coefficient are reported.

Table 1. Machine learning methods that ignore location and time when applied to spatial-temporal phenomena.

	Target	Features	Machine learning method	Accuracy
[6]	Classification with two classes: Propensity of forests in southern	• Distance to the nearest road weighted by the average transportation cost	Logistic regression (different models are developed for	Precision = 89%

	Cameroon for deforestation	• Distance to the nearest market town weighted by the average transportation cost and the price of the agricultural products at the market town • Soil aptitude for agriculture • Shortest distance to the nearest forest/nonforest edge • Spatial fragmentation of the forest cover in the immediate surroundings of each location	different time scales)	
[28]	Classification with 9 classes: Landuse	• Bands 2, 4, 5, and 7 of Landsat TM data • Geology • Hydrology (flow accumulation) • Surface morphology (slope, aspect)	MLP	overall accuracy = 72.61%
[29]	Classification with 6 classes: Landuse	• All four bands of SPOT 6 image (blue, green, red, and near-infrared) • A cluster number assigned to each pixel based on Fuzzy k-means clustering algorithm from all four bands of the image • NDVI calculated for each pixel from the red and near-infrared bands	SVM with Gaussian kernel (using the one-against-all scheme). A 3 × 3 pixel majority filter was applied to all classifications to eliminate the salt and pepper noise.	Overall accuracy = 98%
[30]	Classification with 13 classes: Landuse	Global 8 km resolution AVHRR Pathfinder Land data for 1984 with 24 metrics including, the maximum annual, minimum annual, mean annual, and, amplitude (maximum minus minimum) for • the normalized difference vegetation index (NDVI), • Channel 1 (visible reflectance, 0.58–0.69 μm), • Channel 2 (near-infrared reflectance, 0.725–1.1 μm), • Channel 3 (thermal infrared, 3.55–3.93 μm), • Channel 4 (thermal, 10.3–11.3 μm), and • Channel 5 (thermal, 11.5–12.5 μm)	Decision tree	Overall accuracy = 85%
			Decision tree with bagging	Overall accuracy = 87%
			Decision tree with boosting	Overall accuracy = 89.5%
	Classification with 6 classes: Landuse	Landsat Thematic Mapper scene around Pucallpa, Peru acquired 16 October 1996 including • five bands at 30 m resolution (.45–.53 μm, .52–.60 μm, .63–.69 μm, .76–.90 μm, and 1.55–1.75 μm)	Decision tree	Overall accuracy = 84.5%
			Decision tree with bagging	Overall accuracy = 87%
			Decision tree with boosting	Overall accuracy = 89.5%
[31]	Classification with 5 classes: Grassland type	• SAR data (a total of 12 ENVISAT ASAR images operated at C-band, 15 ERS-2 images operated at C-band, and 12 ALOS PALSAR images operated at L-band), • Ancillary data (soils, sub-soils, elevation, and slope)	SVM with Gaussian kernel (using the one-against-one scheme)	Overall accuracy = 92.5- 97.9%
			RF	Overall accuracy = 92.4-98%
			Extremely randomized trees	Overall accuracy = 94.1- 98.7%
[32]	Classification with 2 classes: Miscanthus presence/absence at the level of the farmer's block	• Agronomical variables (topsoil water capacity, soil texture, and distance to rivers) • Morphological variables (size, shape, and maximum values of elevation and slope for each farmer's block) • Contextual variables (the farmer's block distance to the overall farmland, to the transformation plant, and to the road, proximity to the built-up areas, and length of the parcel boundaries shared with the neighboring woods)	Boosted regression tree	AUC (for training data) = 0.793
[10]	Classification with 2 classes: Presence or absence of harmful algal blooms in the Gulf of Mexico	• The level-2 SeaWiFS sensor data (bands at 443, 490, 510, and 555 nm, and Chlorophyll-a) for the period 1999–2004 with a spatial resolution of 1.1 Km. • The level-2 MODIS-A sensor data (bands at 412, 443, 488, 531, and 551 nm and Chlorophyll-a) for the period 2002–2004 with a spatial resolution of 1 Km. • Ancillary data (meteorological and ozone data)	SVM with HTRBF kernel (the classifier's parameters are optimized trough cross validation with Genetic	Kappa coefficient = 0.75

		• The feature vector (spectral data) at each sample is the average of features in a cubical window of size 3° of latitudes, 3° of longitudes, and 3 days centered at that sample. • Kernel PCA was used to transform features and only the first 300 components for SeaWiFS features and 72 components for MODIS-A features are used.	Algorithm applied to narrow down the search space)	
[33]	Classification with 2 classes: Landslide prone/non-prone areas	• Convergence index (characterizes soil and debris erosion and deposition within the landscape) • Compound terrain index (the logarithm of the ratio between upslope contributing area and slope gradient) • Distance from channel base level (the limiting level below which a stream cannot erode its channel) • Distance from faults • Distance from thrust • Downslope distance gradient (how far a given amount of water must travel in the landscape to lose a certain amount of potential energy)	v-SVM with radial basis function Gaussian kernel (the classifier's parameters are optimized trough cross validation)	AUC = 0.83
		• Elevation • Insolation (the amount of radiation reflected by the terrain) • Internal relief (maximum elevation change per unit area)	Logistic regression	AUC = 0.79
		• Morphological protection index (the positive openness which expresses the degree of dominance or enclosure of a location within the landscape) • Slope • Stream power index (expresses the erosive potential of overland flow)	Linear discriminant analysis	AUC = 0.79
		• The presence of clay and marl-rich sedimentary formations • Attitude of rock strata • Landuse	Naïve Bayes	AUC = 0.76
[34]	Regression: Intra-and-inter species forest aboveground biomass level	• All five bands of 5m RapidEye images (blue, green, red, near infrared, and red-edge) • NDVI and 13 other vegetation indices calculated for each pixel from different bands of RapidEye images	Stochastic gradient boosting regression tree	$R^2 = 0.61$
			RF	$R^2 = 0.37$

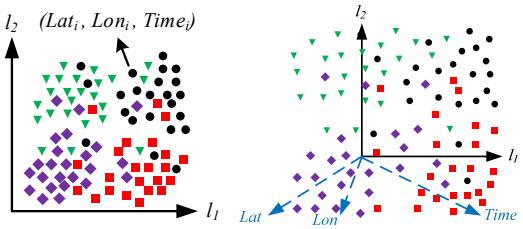


Fig. 1. The recorded location and time of training samples are ignored (left), location and time are considered as additional features (right).

Fig. 1 shows a schematic spatial-temporal training dataset with four patterns and two features. Current machine learning techniques treat spatial-temporal problems no differently than other types of problems [35]. They do not take spatial and/or temporal autocorrelations into account, neither in training nor in testing the predictor. That results in poor performance of machine learning techniques in the presence of spatial-temporal data [13,14,15]. On the other hand, considering location and time as features in the training process is not the best way to incorporate the result of autocorrelation [17]. The

reason is that, it leaves autocorrelated prediction residuals behind [18,19,26] and increases the sparseness of training samples in the feature space, as schematically shown in Fig. 1. It also increases the number of free parameters in the predictor and consequently the demand for larger training datasets, referred to as the curse of dimensionality [36].

3 Combining Inductive and Analytical Learning

Spatial autocorrelation can be measured using semivariance (γ). Equation 1 [1] shows how semivariance for lag d is calculated, where n_d is the number of observation pairs with a distance (d_{ij}) between d-$\Delta/2$ and d+$\Delta/2$, Δ is the lag interval, and y_i and y_j are the observations i and j, respectively.

$$\gamma(d) = \frac{1}{2n_d} \sum_{d_{ij}=d-\Delta/2}^{d+\Delta/2} \left(y_i - y_j\right)^2 \qquad (1)$$

Observations that are geographically near each other are more likely to be similar than distant observations. This fact results in an increase in semivariance with spatial lag (d) shown schematically in Fig. 2.

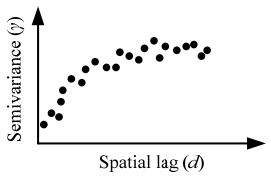


Fig. 2. Semivariance versus spatial distance.

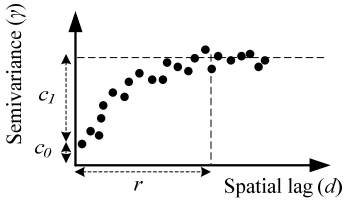


Fig. 3. Nugget (c_0), partial sill (c_1), and range (r) in a semivariance versus spatial distance plot.

Sill (c_0+c_1) is the semivariance upper bound (see Fig. 3). Partial sill (c_1) is the sill minus the nugget and is measured as the limit of spatial semivariance as the spatial distance approaches infinity. This is estimated by the variance of observations (σ^2). Nugget (c_0) presents a discontinuity in the semivariance at the origin (see Fig. 3). It is the semivariance at an infinitesimally small lag. The range (r) is the lag at which the semivariance flattens out (see Fig. 3).

Many empirical spatial semivariograms approximate to a spherical model shown in Equation 2 and visualized in Fig. 4. The spherical model is the most frequently used model, although exponential and Gaussian models are two other common semivariogram models [37]. They are shown in Fig 4 as well and calculated via Equations 3 and 4 for exponential and Gaussian models, respectively. All three models depend only on nugget (c_0), partial sill (c_1), and range (r).

$$\hat{\gamma}(d_s) = \begin{cases} c_0 + c_1 \left[\frac{3d_s}{2r} - 0.5\left(\frac{d_s}{r}\right)^3\right] & d_s \le r \\ c_0 + c_1 & d_s > r \end{cases} \qquad (2)$$

$$\hat{\hat{\gamma}}(d_s) = c_0 + c_1\left[1 - \exp\left(\frac{3d_s}{r}\right)\right] \quad (3)$$

$$\hat{\hat{\gamma}}(d_s) = c_0 + c_1\left[1 - \exp\left(\frac{3d_s^2}{r^2}\right)\right] \quad (4)$$

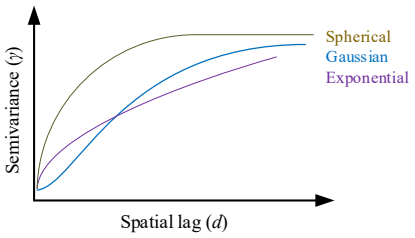


Fig. 4. Spherical, Gaussian, and exponential semivariogram models fitted to the semivariances in Fig. 1.

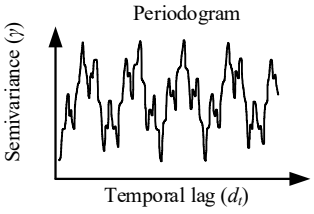


Fig. 5. Semivariance versus temporal distance.

The semivariogram can also be used to model the autocorrelation among observations over time rather than space. The shape of temporal semivariogram however will not necessarily be the same as spatial semivariogram. The autocorrelation among observations is more complicated over time than space because not only temporally closer observations are more likely to be similar than temporally farther observations, but also, they might exhibit multiple periodic behaviors, with different frequencies and amplitudes, over time as shown in Fig. 5.

Fig. 6 shows observations in the space-time domain. Spatial and temporal autocorrelations are schematically shown in this figure, where samples close to each other in the space-time domain are similar and also repeat themselves periodically over time. Fig. 7. schematically shows the spatial-temporal similarity of training samples with respect to a test sample in the space-time domain, while Fig. 8 shows the same in the feature space.

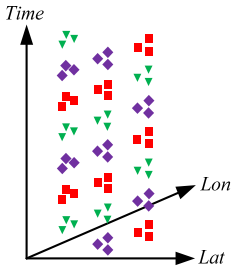


Fig. 6. Spatial-temporal data in the space-time domain.

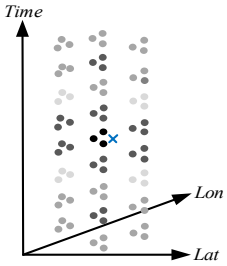


Fig. 7. Training samples shaded based on their spatial-temporal similarity with a test sample (blue cross) in the space-time domain.

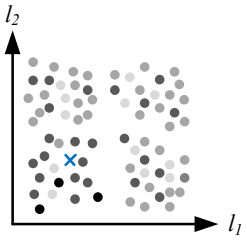


Fig. 8. Training samples shaded based on their spatial-temporal similarity with a test sample (blue cross) in the feature space.

Inductive learning is a statistical approach that applies training samples to discover the true hypothesis. The three strategies used in the machine learning literature – ignoring location and time, using location and time as the only features, and considering location and time as features in addition to other features – to handle spatial-temporal data fall in this category. Analytical (or deductive) learning is a logical approach that applies prior knowledge or domain theory to derive the true hypothesis [38]. This is if a prediction is made by solely applying the special characteristics of spatial-temporal phenomena and no training samples. One potential solution to our problem is to combine inductive and analytical learning, as shown in Fig. 9. We have some training samples and some prior knowledge or domain theory, both imperfect and erroneous. We can combine inductive and analytical learning, to take advantage of both and derive a more accurate hypothesis [39].

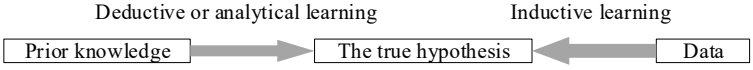


Fig. 9. Combining inductive and analytical learning.

To this aim, first a weight is assigned to each training sample and then weighted machine learning models are used to make predictions for spatial-temporal data. This approach is superior to existing machine learning models, theoretically because it captures the spatial-temporal autocorrelation properly, and practically as our experiments with real data indicate.

To assign a weight to a training sample, first its distance to the test sample is measured, the semivariance for this distance is found from the spatial semivariogram model, and its inverse is used as that training sample's weight. A similar approach is used to assign a weight to other training samples. It is noteworthy that this is a lazy training, since the training samples' weights depend on the test sample's location. Temporal autocorrelation is not treated here, but will be incorporated in future research.

We will apply the weighted machine learning models developed by Hashemi and Karimi [40] to take advantage of the calculated weights in training the machine. In their weighted machine learning models, misclassifying training samples with higher weights is costlier and the predictor will attempt to avoid it. In other words, training the weighted predictor is more concerned with correct prediction of training samples with larger weights than those with smaller weights.

4 Results and Discussion

The MATLAB software on a 64-bit platform with 8 GB RAM, a Core i7 CPU and a 2.00GHz processor was used in experiments. A real world application is presented in this section: regression of air temperature based on meteorological features. A classification problem with simulated data is also presented afterwards. The overall accuracy is reported for the classification problem and RMSE and R^2 are reported for the regression problem.

Oceanographic and surface meteorological readings, taken from a series of buoys positioned throughout the equatorial Pacific, from 1980 to 1998, are available to public in [41]. Each reading includes location, date, zonal winds, meridional winds, humidity, and temperature. The dataset has 178,000 records. Removing records with missing features leaves the dataset with 94,000 records. In this experiment, we will predict the air temperature based on zonal winds, meridional winds, and humidity.

We applied linear LS and parametric Bayesian regressor with Parzen windows and a Guassian kernel [42] for this regression problem. In addition to our proposed approach, three different scenarios were cross-tested using leave-one-out approach: (a) location and time were ignored, (b) location and time were considered as the only features, and (c) location and time were considered as features in addition to zonal winds, meridional winds, and humidity. The results are reported in Table 2.

Table 2. Accuracy of regression techniques for predicting the temperature.

Technique	RMSE	R^2
LS with location and time as the only features	1.62	0.07
LS without location and time as additional features	1.37	0.33
LS with location and time as additional features	1.35	0.35
Proposed methodology with weighted LS	**0.96**	**0.67**
Bayesian regressor with location and time as the only features	1.40	0.30
Bayesian regressor without location and time as additional features	1.31	0.39
Bayesian regressor with location and time as additional features	1.07	0.59
Proposed methodology with weighted Bayesian regressor	**0.92**	**0.70**

In the second experiment, we generated 10,000 random samples from each of the two classes A and B. Samples are pulled from two five-dimensional normal probability distribution functions with the following mean vectors (μ_A and μ_B) and symmetric positive-definite covariance matrixes (Σ_A and Σ_B):

$$\mu_A = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \Sigma_A = \begin{bmatrix} 10 & 0 & 1 & 1 & 2 \\ 0 & 10 & 3 & 4 & 1 \\ 1 & 3 & 10 & 2 & 3 \\ 1 & 4 & 2 & 10 & 4 \\ 2 & 1 & 3 & 4 & 10 \end{bmatrix} \mu_B = \begin{bmatrix} 101 \\ 101 \\ 101 \\ 101 \\ 101 \end{bmatrix} \Sigma_B = \begin{bmatrix} 10 & 1 & 2 & 1 & 4 \\ 1 & 11 & 4 & 1 & 3 \\ 2 & 4 & 10 & 2 & 2 \\ 1 & 1 & 2 & 11 & 4 \\ 4 & 3 & 2 & 4 & 10 \end{bmatrix}$$

Location is defined for samples to satisfy a spherical semivariogram with the following characteristics: c_1=0.3, c_0=0, and r=25. Time is defined for samples to satisfy a single-frequency periodogram with the following characteristics: amplitude=0.05, period=20, and c_0=0. A white noise was also added to location and time of training samples.

We applied LS, SVM, MLP, decision tree (DT), and parametric Bayesian classifier with Parzen windows and a Guassian kernel [42] for this classification problem. In addition to our proposed approach, three different scenarios were again cross-tested using leave-one-out approach: (a) location and time were ignored, (b) location and time were considered as the only features, and (c) location and time were considered as features in addition to other features. The results are reported in Table 3.

Table 3. The accuracy of different classification techniques for the simulated data.

Technique	Overall Acc. %
LS with location and time as the only features	56.95
LS without location and time as additional features	56.03
LS with location and time as additional features	56.39
Proposed methodology with weighted LS	**97.52**
Bayesian with location and time as the only features	53.66
Bayesian without location and time as additional features	46.74
Bayesian with location and time as additional features	57.17
Proposed methodology with weighted Bayesian classifier	**70.14**
DT with location and time as the only features	51.18
DT without location and time as additional features	45.59
DT with location and time as additional features	51.52
Proposed methodology with weighted DT	**96.36**
SVM with location and time as the only features	49.53
SVM without location and time as additional features	45.98
SVM with location and time as additional features	48.31
Proposed methodology with weighted SVM	**62.57**
MLP with location and time as the only features	50.58
MLP without location and time as additional features	44.11
MLP with location and time as additional features	45.42
Proposed methodology with weighted MLP	**96.48**

It is noteworthy that we are only concerned about how different strategies of dealing with location and time would affect the results, rather than comparing different machine learning techniques. In both experiments, the proposed methodology reached a higher accuracy. In the first experiment, considering location and time as additional features, revealed the second-best accuracy, followed by ignoring them altogether and eventually considering them as the only features.

In the second experiment, ignoring location and time altogether resulted in the lowest accuracy. In case of Bayesian classifier, considering location and time as additional features resulted in a higher accuracy than considering them as only features. This result was reversed in case of MLP. For other classifiers, the difference between the accuracy of these two strategies, i.e. considering location and time as additional features or only features, remained less than 1%.

5 Conclusions and Future Directions

Three strategies are used in the literature when machine learning is applied to spatial-temporal data: (a) ignoring location and time altogether and training the machine using other features, (b) using location and time as the only features and ignoring other features when training the machine, and (c) considering location and time as features in addition to other features. The second approach achieves good accuracies in experiments and the third approach seems the most reasonable way of exploiting all the

available information. However, all three solutions leave important traits of spatial-temporal data behind and thereby reduce the training process's effectiveness. In this study, we identified the characteristics of spatial-temporal data that make them different than other types of data and also affect the training process. We proposed a methodology to embed the external knowledge about spatial autocorrelation into weighted machine learning techniques. While we theoretically showed that the proposed approach captures the spatial autocorrelation more precisely, this was also confirmed by its higher accuracy in experiments with two datasets. A future research venue is to take the temporal autocorrelation into account as well.

References

1 O'Sullivan D, Unwin D. Geographic information analysis. Hoboken, New Jersey: Wiley; 2010.

2 Hashemi M, Alesheikh A. Spatio-temporal analysis of Tehran's historical earthquakes trends. In: Advancing Geoinformation Science for a Changing World. Springer; 2011. p. 3-20.

3 Hashemi M, Karimi HA. Seismic source modeling by clustering earthquakes and predicting earthquake magnitudes. In: Smart City 360°. Springer; 2016. p. 468-478.

4 Hashemi M. Reusability of the Output of Map-Matching Algorithms Across Space and Time Through Machine Learning. IEEE Transactions on Intelligent Transportation Systems. 2017a;18(11):3017-3026.

5 Hashemi M, Karimi HA. A machine learning approach to improve the accuracy of GPS-based map-matching algorithms. In: In Proceedings of the IEEE 17th International Conference on Information Reuse and Integration; 2016b; Pittsburgh, PA. p. 77-86.

6 Mertens B, Lambin EF. Land-cover-change trajectories in southern Cameroon. Annals of the Association of American Geographers. 2000;90(3):467-494.

7 Lucas RM, Honzak M, Foody GM, Curran PJ, Corves C. Characterizing tropical secondary forests using multi-temporal Landsat sensor imagery. International Journal of Remote Sensing. 1993;14(16):3061-3067.

8 Alves DS, Skole DL. Characterizing land cover dynamics using multi-temporal imagery. International Journal of Remote Sensing. 1996;17(4):835-839.

9 Tucker CJ, Dregne HE, Newcomb WW. Expansion and contraction of the Sahara Desert from 1980 to 1990. Science. 1991;253:299-301.

10 Gokaraju B, Durbha SS, King RL, Younan NH. A machine learning based spatio-temporal data mining approach for detection of harmful algal blooms in the Gulf of Mexico. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2011;4(3):710-720.

11 Hashemi M, Alesheikh A, Zolfaghari MR. A GIS-based time-dependent seismic source modeling of Northern Iran. Earthquake Engineering and Engineering Vibration. 2017;16(1):33-45.

12 Hashemi M, Alesheikh AA, Zolfaghari MR. A spatio-temporal model for probabilistic seismic hazard zonation of Tehran. Computers & Geosciences. 2013;58:8-18.

13 Santibanez S, Kloft M, Lakes T. Performance analysis of machine learning algorithms for regression of spatial variables. A case study in the real estate industry. In: In Proceedings of the GeoComputation Conference; 2015a; Dallas, Texas.

14 Santibanez S, Lakes T, Kloft M. Performance analysis of some machine learning algorithms for regression under varying spatial autocorrelation. In: AGILE; 2015b; Lisbon.

15 Fotheringham AS, Brunsdon C, Charlton M. Geographically weighted regression. Chichester, England: Wiley; 2002.

16 Franklin J. Predictive vegetation mapping: geographic modelling of biospatial patterns in relation to environmental gradients. Progress in Physical Geography. 1995;19(4):474-499.

17 Shekhar S, Evans MR, Kang JM, Mohan P. Identifying patterns in spatial information: A survey of methods. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2011;1(3):193-214.

18 Kanevski M, Parkin R, Pozdnukhov A, Timonin V, Maignan M, Demyanov V, Canu S. Environmental data mining and modeling based on machine learning algorithms and geostatistics. Environmental Modelling & Software. 2004;19(9):845-855.

19 Gilardi N, Bengio S. Comparison of four machine learning algorithms for spatial data analysis. In: Dubois G, Malczewski J, Cort MD. Mapping radioactivity in the environment: spatial interpolation comparison 97. Luxembourg: Office for Official Publications of the European Communities; 2003. p. 222-237.

20 Dubin RA. Predicting house prices using multiple listings data. Journal of Real Estate Finance and Economics. 1998;17(1):35-59.

21 Li J, Heap AD, Potter A, Daniell JJ. Application of machine learning methods to spatial interpolation of environmental variables. Environmental Modelling & Software. 2011;26(12):1647-1659.

22 Breiman L. Random forests. Machine Learning. 2001;45(1):5-32.

23 Smola AJ, Schölkopf B. A tutorial on support vector regression. Statistics and Computing. 2004;14(3):199-222.

24 Kanevski M, Timonin V, Pozdnoukhov A. Automatic Mapping and Classification of Spatial Environmental Data. In: Geocomputation, Sustainability and Environmental Planning. Springer; 2011. p. 205-223.

25 Parzen E. On estimation of a probability density function and mode. The annals of mathematical statistics. 1962;33(3):1065-1076.

26 Gilardi N, Bengio S. Local machine learning models for spatial data analysis. Journal of Geographic Information and Decision Analysis. 2000;4(1):11-28.

27 Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE. Adaptive mixtures of local experts. Neural Computation. 1991;3(1):79-87.

28 Gahegan M, German G, West G. Improving neural network performance on the classification of complex geographic datasets. Journal of Geographical Systems. 1999;1(1):3-22.

29 He T, Sun YJ, Xu JD, Wang XJ, Hu CR. Enhanced land use/cover classification using support vector machines and fuzzy k-means clustering algorithms. Journal of Applied Remote Sensing. 2014;8(1).

30 DeFries RS, Chan JCW. Multiple criteria for evaluating machine learning algorithms for land cover classification from satellite data. Remote Sensing of Environment. 2000;74(3):503-515.

31 Barrett B, Nitze I, Green S, Cawkwell F. Assessment of multi-temporal, multi-sensor radar and ancillary spatial data for grasslands monitoring in Ireland using machine learning approaches. Remote Sensing of Environment. 2014;152:109-124.

32 Rizzo D, Martin L, Wohlfahrt J. Miscanthus spatial location as seen by farmers: A machine learning approach to model real criteria. Biomass and Bioenergy. 2014;66:348-363.

33 Ballabio C, Sterlacchini S. Support vector machines for landslide susceptibility mapping: the Staffora River Basin case study, Italy. Mathematical Geosciences. 2012;44(1):47-70.

34 Dube T, Mutanga O, Elhadi A, Ismail R. Intra-and-inter species biomass prediction in a plantation forest: testing the utility of high spatial resolution spaceborne multispectral rapideye sensor and advanced machine learning algorithms. Sensors. 2014;14(8):15348-15370.

35 Hashemi M, Karimi HA. Weighted machine learning. Statistics, Optimization and Information Computing. 2018;6(4):497-525.

36 Evangelista PF, Embrechts MJ, Szymanski BK. Taming the curse of dimensionality in kernels and novelty detection. In: Applied soft computing technologies: the challenge of complexity. Berlin: Springer; 2006. p. 425-438.

37 Bohling G. Introduction to geostatistics and variogram analysis. Kansas geological survey. 2005;2.

38 Hashemi M. Automatic inference of road and pedestrian networks from spatial-temporal trajectories. IEEE Transactions on Intelligent Transportation Systems. 2019 DOI: 10.1109/TITS.2019.2916588.

39 Mitchell TM. Machine learning. McGraw Hill; 1997.

40 Hashemi M, Karimi HA. Weighted machine learning. Statistics, Optimization and Information Computing. 2018;6(4):497-525.

41 Oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. Pacific Marine Environmental Laboratory , National Oceanic and Atmospheric Administration , US Department of Commerce. Available from: http://www.pmel.noaa.gov/.

42 Theodoridis S, Koutroumbas K. Pattern Recognition. 4th ed. Elsevier; 2009.

Detecting Periodic Correlated Hashtags in Twitter

Sultan Alzahrani[0000−0001−8931−1558], Saud Alashri[0000−0002−8114−0130], Manal Alhassoun, Sara Alghunaim, Sarah Alasraj, and Muneera Alhoshan[0000−0001−5705−3590]

National Center for AI and Big Data Technologies
King Abdulaziz City for Science and Technology (KACST)
Riyadh, Saudi Arabia
{szahrani, salashri, malhassoun, salghunaim, salasraj, malhawshan}@kacst.edu.sa

Abstract. Detecting correlated hashtags and topics in social media aim to provide topic awareness by uncovering the "Unknown" topics to be "known" at their nascent stages. In this paper, we propose a framework to detect correlated hashtags and topics on Twitter. Our dataset is comprised of tweets collected from Saudi Arabia as it contributes large amounts of hot topics regionally and scored the highest social media penetration globally. In this framework, we merge scatter related hashtags representing topics through hashtag co-occurrence. Then, we identify the main hashtag per cluster that is representative of its topic. Since not all of the tweets have hashtag and in order to not lose a valuable information from these tweets, we utilize Non-negative Matrix Factorization (NMF) to extract topics from the remaining data. The model was evaluated quantitatively and qualitatively and benchmarked against both Latent Dirichlet Allocation (LDA) and NMF. The proposed model achieved the highest topic coherence over various measures, and the lowest $Hscore$ equal to 0.353 corresponding to the best cohesive outcome topics among other models. Also, through manual evaluation, it yielded the highest precision, with an average of 0.85 (out of 1.0). These findings suggest that the proposed model can be a utility in correlated hashtags detection and clustering, which help in summarizing large amounts of tweets and getting precise insights.

Keywords: Correlated Hashtags Detection · Text Processing · Text Analysis · Natural Language Processing · Social Media.

1 Introduction

Detecting coherent topics and trends automatically in communities looming in social media is one of the most challenging tasks in Natural Language Processing and Machine Learning arena. It aims to create "Topic" awareness and to address a challenging problem of converging the "Unknown" topics that start evolving and diffuse within and across communities of users to become "Known" coherent

and readable. Correlated Hashtags and Topic detection serves a broad spectrum of applications such as politics[28], economics [10], and public health [17] to find topics correlated with trends and events that are unknown beforehand. Such patterns of social processes reflected in Social Networking Sites (SNS) necessitate analysts to be "the head of the curve" given the fact the curse of vast volumes of contents generated from these outlets.

Due to the immense amount of data produced by SNS, manual detection of topics is costly and not scalable. As such, machine learning techniques overcome these limitations by automating content analysis. Many studies have addressed topic detection using topic modelling approaches such as Latent Semantic Analysis (LSA) [13], LDA [4] and NMF [8]. Such approaches may perform well on long-form text articles, but their performance significantly decreases with poor quality of interpretable topics when applied on short text such as Tweets due to sparseness issue. [24,14,31].

The initial efforts for topic detection relied on employing LSA, NMF, LDA, and others employ unsupervised learning approach. In this paper, we are proposing a novel approach that semantically merges similar hashtags based on their co-occurrence with an expansion approach that overcomes sever sparsity in short texts. The prime focus of this paper is on the content of Arabic tweets in Saudi dialect due to its undergone rapid growth in recent years, representing the largest 99% social media penetration in the world for 2019 [6]. Both qualitative and quantitative experimentation shows a significant improvement, and higher quality returned a set of topics when benchmarked against LDA and NMF.

The rest of this paper is organized as follows. The next section provides related work. Section 3 describes our research methodology. The results and analysis of findings are presented in Section 4. Section 6 concludes the paper and outlines opportunities for future research.

2 Related Work

Topic modeling and their enhanced models in detecting events in Twitter is still a growing research area due to the massive information flow with short and noisy content[18]. This section surveys some existing studies conducted in the field of modeling and detecting topics on Twitter. The majority of previous work focuses on real-life or social events detection. Additionally, in the expanse of Arabic tweets, few studies have been conducted on topic modeling.

Ozdikis et al.[22] and Ma et al.[18] proposed detection methods based on hashtags. Ozdikis et al.[22] detect events on Twitter by clustering the hashtags, rather than entire tweet content, and propose an improvement technique through utilizing semantic similarities between the hashtags. They experimented using hashtag-based tweet vector with and without semantic expansion. They utilized hashtags similarities scores during the lexico-semantic expansion on tweet contents before the clustering process. The results revealed that hashtag-based tweets representation benefits the overall performance than the word-based tweet vectors in term of coverage and quality of the obtained clusters.

Ma et al. [18], attempted to predict the popularity of Twitter topics based on hashtags using five different classification algorithms including Naïve Bayes, k-nearest neighbors, decision trees, support vector machines, and logistic regression. During the classification, they consider only two types of features: context-based and content-based. Contextual features are those that are extracted from Twitter users' social graphs, whereas the content-based features are derived from tweets' content containing the hashtag as well as the hashtag string lexically. Their findings suggest the effectiveness of contextual features over contentbased features.

A semi-automatic training approach to detect tweets reported on social events [15]. They argued that their work reduces the efforts of manual labeling training datasets. They build a classifier that classifies tweeter messages into two mutually exclusive groups: Event-related and Non-event related tweets. They employed Naïve Bayes classication algorithm on content published by selected event broadcasters with the assumption that broadcasting only on event-related tweets. The algorithm utilizing n-gram features extracted from broadcasters' Twitter content and their followers. The process of assigning an events class is based on applying a heuristic rule that defines the appearance of three different aspects related to the event including time, persons involved and locations. The tweets posted by other users are classified under other class. They found that this semi-automated method achieves higher accuracy compared to other manual labeling approaches. Although this method could be used as a filtering step before applying other event-related content detection techniques, they rely on the assumption of tweeting about related events by the broadcasters. Besides, their approach utilizes only one type of feature, as well as its evaluation does not employ other classification models as presented in [18].

For the Arabic language, few research has been conducted compared to other languages on this field, mainly due to the variety of Arabic regional dialects and the lack of Modern Standard Arabic (MSA) use on social media [3].

Alsaedi et al. [1,2] proposed a hybrid approach for detecting unknown distributive events on Twitter, where both classification and clustering techniques were applied. The dataset collected from Twitter stream using, keywords, user's profile location, and geographic hashtags. The dataset includes a variety of Arabic tweets from different Arab countries. The detection process goes through several stages. The approach, first, binary classifies tweets into "event" and "nonevent" in a way to reduce the amount of noisy nonevents tweets. Then, they proceeded with clustering the rest of event tweets into topics followed by summarizing each cluster and showing the most important keywords per cluster. In [2], they used a temporal TF-IDF weighting instead of the voting approach. The method they applied depends on different features, such as geographical, temporal, textual, and some other metadata of tweets. The efficiency of this approach depends on the quality of the classification process that is trained by relatively small static dataset, which we deemed not suitable for Twitter due to its thematic diversity. Thus it can hardly generalize.

A feature-pivot method developed by [11]. They provided a technique for bursty event detection from Twitter stream. Moreover, this technique analyzes the tweet's text to discover any abnormal pattern in specific timeframe. The tweets collected form Egypt and divided into chunks based on their post time. These chunks were processed sequentially using TF-IDF and entropy measures. The method was evaluated against detecting up to three already known events on that time period. The results showed that approach is able to capture bursty features of real events.

On the other hand, a document-pivot approach is studied by [25]. They compared different clustering techniques, namely, k-means, repeated bisecting k-means, agglomerative, and biased agglomerative. These techniques were applied on Egyptian dialect tweets on the basis of their similarity. Furthermore, multiple hashtags extracted from each cluster and considered as trending topics based on their occurrences. For the evaluation, they tested the extracted hashtags against annotated tweets. The best results were achieved with the repeated bisecting k-means clustering.

Our contribution is different from the previous works. We present a framework that utilizes generated hashtags associated with the tweets. Our work can be summarized as follows: 1) capturing the semantic reference between tweets that representing main coherent topics. 2) employing hashtags cooccurrence and Word2vec model [23] that serves to capture common underlying patterns used in merging hashtags. 3) we utilize NMF to extract topics from tweets no pertain to any topics.

3 Methodology

We propose a framework that detect the correlated hashtags and topics by finding semantic relationships between the tweets. In order to do so we follow three main stages: First, we model the tweets in a symmetrical co-occurrence matrix based on the hashtags co-occurrence [19]. Each cell in the matrix represents a frequency value for the pair of hashtags that co-occurred together in the same tweet. Second, merge detected correlation based on the assumption that there are transitive relations between these hashtags. Then, if pair of hashtags are co-occurred and one of these hashtags used with another hashtag,they are considered as related to each other semantically. To achieve this goal, we model the correlated hashtags as a graph, where each node in the graph shows a hashtag and each edge represents the co-occurrence relationship between the two hashtags and the number attached to each edge represents the co-occurrence frequency. Third, for the tweets that are without hashtag we used the NMF to discover their latent topics.

3.1 Hashtag Co-occurrence Topic Model

LDA and NMF are widely used algorithms for topic modeling. However, they are not effective enough in discovering underlying topics from short, sparse, noisy,

and unstandardized text such as tweets [30]. Thus, to address this limitation, we utilize the user-generated hashtags associated with tweets to discover the current popular topics. Hashtags are used to represent and summarize the main idea of tweet's content where users embed posts with a tag to organize, promote content, and draw attention. Other Twitter users, likewise, can find, follow, and contribute in these hashtags. Hence, co-occurring hashtags (i.e. hashtags present in the same tweet) are usually represent the same topic.

The formal generative process of **Hashtag Co-occurrence Topic Model** is as follows:

1. Iterate over all hashtags $H = \{h_1, \ldots, h_k\}$ in the corpus and generate the following triplet for each hashtag:
   $< h_i, tweettext\_lst_i >$
   (1) h_i denotes a hashtag i
   (2) $tweettext\_lst_i$ denotes all tweets posts as lists that includes h_i
2. Build co-occurrence matrix A of hashtags such that a hashtag co-occurred with a hashtag if they both happened to be posted under the same tweet.
3. Any two hashtags co-occurred more often than a certain threshold value is an indication of a semantic relationship between them. Hence, based on the co-occurrence matrix, we determine a suitable minimum frequency as threshold.
4. We assumed that their are transitive relations between the hashtags. Furthermore, if pair of hashtags are co-occurred and one of these hashtags happened to co-occurred with another hashtag then they represent the same coherent topic. To find the transitive relations, we model the co-occurrence matrix as a graph where each node represent a hashtag and each edge represent co-occurrence value. Based on the experimentally determined threshold value, we merge the nodes that have connections to represent one topic (see figure 1).
5. Expansion using Word2vec: Expand each topic by taking the top five most similar words of each topic's hashtags using a Word2vec model and merge topics that share same words after the expansion.
6. Finally, we label each topic based on the most frequent hashtag in all tweets which belong to that topic.

3.2 Non-negative Matrix Factorization (NMF)

From the previous step, some tweets are unlikely to be capable of being part of a topic. For example, tweets with no hashtag, or even tweets with hashtag(s) but neither occur with any other hashtags ,nor they satisfy the threshold. These tweets are grouped under one set, then, their latent topics are determined by applying an alternative approaches such as: LDA and NMF. The generative probabilistic model LDA and the geometrically linear algebra NMF are common techniques for discovering latent topics. Since we are dealing with tweets, the text is very short, noisy, and sparse when compared with other types of long-form documents such as news articles. Although both NMF and LDA needs

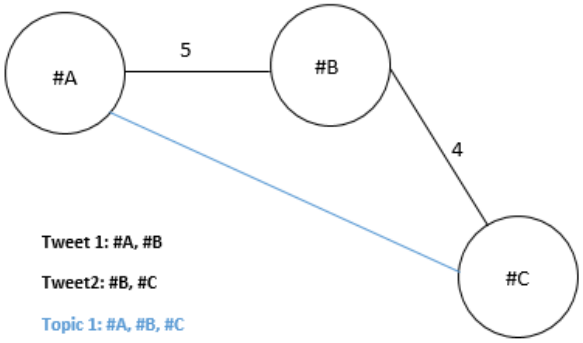


Fig. 1: Graph that represent a transitive relation between hashtags

a better quality of data to produce an interpretable set of topics, the NMF has prior steps before constructing the model that might help in reducing the noise, such as encoding the text as TF-IDF. In this step, the most useful terms (features) from the text are extracted. However, in LDA, there is only TF (Term Frequency) as a representation of data before generating the model. Moreover, NMF leverages multiplicative update algorithm, a definite learning scheme for approximating the original texts' information[7]. On the other hand, the short and noisy text does not provide valuable information about word to word co-occurrences which lead to variance when applying LDA's stochastic Gibbs words sampling. Hence, for the aforementioned reasons, we chose NMF to extract topics from the remaining tweets.

We constructed an NMF model for topic modeling as follows:

1. Extracting features (most indicative terms) from our dataset using TF-IDF by converting a collection of documents into a matrix of TF-IDF features. In this process, we take into account to limit the features corresponding to terms to the one appears at least twice in the entire corpus and at most 95% of the documents
2. Generating the DocumentTerm matrix with the TF-IDF weights
3. Then NMF model is constructed and initialized using non-negative double singular value decomposition (NNDSVD [5]) initialization method which helps in reducing NMF errors and make it more stable
4. The topic terms matrix are given by a matrix H corresponds to terms by topics weights that result from factorizing the TF-IDF matrix into two matrices, W and H, where $A \simeq WH$.
5. For each topic, we extract the most representative term as the topic's primary term which can be either the most frequent hashtag (that already exist in the topic terms) or the term with the highest weight compared to other terms.

4 Experiment

4.1 Data Collection and Preprocessing

Since this study focuses only on tweets posted in Saudi Arabia, we utilized Twitter's stream API through Tweepy Python library [27] to stream tweets that were published from within the country. Even though the location constraint is applied, some tweets appear from other nearby countries such as Kuwait and Yemen. These were eliminated by conducting further filtering step based on location information such as country code and coordinates of the Saudi Administrative Regions.

Furthermore, the manual content analysis for a random number of tweets, reveals that the tweets that compromise more than three hashtags are usually irrelevant to their hashtags and mainly used for advertising or trending the hashtags. Thus, we omitted these tweets in order to avoid their possible negative effect on our result. It also reveals that some of the tweets are auto-generated messages (e.g. Islamic supplications and Foursquare check-ins) that were published using third-party applications (e.g. du3a.org and swarmapp.com). Therefore, the source of the tweets also examined to discard those tweets by monitoring a list of auto-generating applications and searching them in the tweet's metadata. Then, tweets were filtered against list of noisy words. This noisy words list was generated by calculate the unigram and bigram frequencies of words in the tweets. Then, we manually checked them to find out the most frequent phrases that are used in advertisement, spam, or inappropriate tweets which could affect topic modeling results since it constitutes a larger proportion compared with other tweets.

In addition, raw data in tweet content is also preprocessed by removing usernames, URLs, punctuation, non-Arabic words, digits, white- space characters, repeated characters, and Arabic stop-words. This is followed by the Arabic Normalization filter, which has some Arabic-specific rules such as: normalize Hamza forms into one form, removal of Arabic diacritics, and removal of Tatweel (stretching character). These steps left us with a dataset consisting of 58,086 tweets that were posted from Saudi Arabia on July 27, 2019.

4.2 Selecting Model Hyperparameters

Grid search was performed over various hyperparameter settings for each model, and we experimentally found the most optimal settings are as follows: $max\_df = 0.8$ corresponds to ignoring terms that appear more than 80% , $min\_df = 2$ corresponds to that appear less than twice , number of grams: $n\_gram = 1$, number of topics $n\_topics = \{10, 20, 30\}$, the minimum threshold of cooccurrence hashtags is experimentally set to be 3.

5 Evaluation

We evaluated the models both quantitatively and qualitatively as follows:

5.1 Topic Coherence

Topic coherence measures the semantic similarity between the top N words in each model-generated topic. For our evaluations, we consider three measures of coherence:

- UMass equation below [21] computes the sum of pairwise score to topic's top N words. Word cooccurance frequencies are estimated based on the original corpus used to train the topic model:

$$UMass = \sum_{(w_i,w_j)\in W} log\frac{D(w_i,w_j)+1}{D(w_i)}$$

  Where $W = w_i, w_j$ the topic's top N words, $D(w_i)$ the document frequency of word w_i, and $D(w_i, w_j)$ count the number of documents in which both wi and w_j occurred.

- UCI [20]: On the contrary to the UMASS, UCI word probabilities are estimated based on document frequencies of external corpus. Using external corpus such as Wikipedia is not suitable to our case due to the informal language used in Twitter, therefore we used a dataset that contains 639,291 Saudi tweets that were streamed from Twitter from July 17-26, 2019 as external corpus. This dataset was preprocessed in similar manner as in section 4.1:

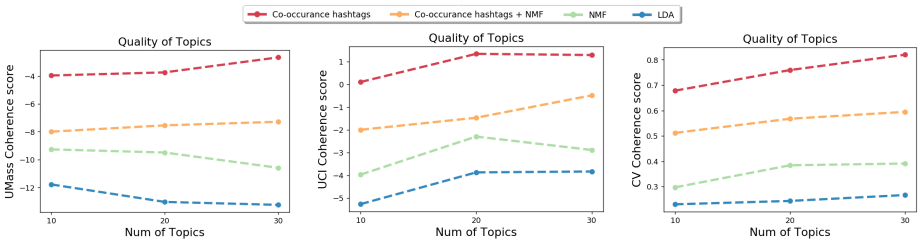
$$UCI = \sum_{(w_i,w_j)\in W} log\frac{p(w_i,w_j)}{p(w_i)p(w_j)}$$

  Where $W = w_i, w_j$ the topic's top N words, $p(w_i)$ the probabilities of seeing w_i in documents, and $p(w_i, w_j)$ the probabilities of seeing both w_i and w_j in same documents.

- CV [26]: In addition to the aforesaid commonly used topic coherence metrics, we consider CV which has been proven for being very appropriate for topic coherence evaluation compared with other models [26]. This metric integrates the normalized point-wise mutual information (NPMI) with the cosine measure, which estimates how similar the topic's top N words co-occurrence patterns in a corpus within a fixed width window.

To evaluate the overall quality of the topic modeling results, we calculate the average coherence score for each method. Considering our model "Co-occurrence Hashtags" generates a maximum of 30 topics for the selected dataset, we evaluate topic models with three different reasonable number of topics for interpretation of topics : 10, 20, and 30. Figures 2a, 2b, and 2c show the coherence scores using the metrics above. From the figures, it is clear that our first proposed model, co-occurrence hashtags, achieved the highest average over topics of -3.453, 0.913, and 0.753 for UMass, UCI, and CV, respectively. These results followed by the

second proposed model, co-occurrence hashtags of values −7.28, −1.316, and 0.558. NMF and LDA scores are the lowest, where NMF scored −0.9777, −3.049 and 0.357, which is relatively higher than LDA scores of −12.683, −4.322 and 0.246. These results suggest that the two proposed models produced more coherent topics in comparison to the NMF and LDA models across the three metrics, namely, the UMass, the UCI, and the CV cohesion.



(a) The UMass coherence scores for each topic model

(b) The UCI coherence scores for each topic model

(c) The CV coherence scores for each topic model

Fig. 2: On all accounts of measures, co-occurrence hashtags model outperforms others

5.2 Model Quality

In this evaluation section, we aim at measuring the models quality by comparing the inter-cluster and intra-cluster distance between models. Each model cluster the tweets into multiple clusters whereby each cluster represents a topic. Tweets context within one cluster hypothetically share an identity topic and thus close to each other and conversely differ from other clusters' tweets. In the NMF and LDA, tweets can associate with all clusters but with different weights or probabilities (i.e., fuzzy clustering). In our evaluation, we map each tweet into the topic/s of the highest probability or the highest weight relative to other topics.

We started by curating a sample collection of tweets by seeding various hashtags and non-hashtags keywords (knowing their topics beforehand) into the Twitter search API such as: "الترفيه/Entertainmet", "الحج/Pilgrimage". Each group of keywords is assigned to a pre-defined topic. Then, we applied our method "co-occurrence hashtags," NMF and LDA, so that, each model generates several topics. In order to calculate the distance between the tweets, each one is represented as a vector of unigrams. To overcome the sparsity and the noisy problem of short text, we represent topics and their keywords in a vector space model (VSM) and weighted by using TF-IDF. Then we utilize the Truncated SVD[12] on the resulted VSM matrix to reduce its dimension. Finally, we

calculate the distance between every two tweets using the Euclidean distance . By using these measures, we consequently measure the overall intra-distance between tweets of the same topic (cluster) and the inter-distance between tweets in different clusters by applying the following equations[30]:

- Average IntraCluster Distance:

$$IntraDis(C) = \frac{1}{K}\sum_{k=1}^{K}\left[\sum_{d_i,d_j \in C_k}\frac{2dis\left(d_i,d_j\right)}{|C_k|\,|C_k-1|}\right]$$

- Average Inter-Cluster Distance:

$$InterDis(C) = \frac{1}{K(K-1)}\sum_{C_k,C_{k'} \in C_{k\neq k'}}\left[\sum_{d_i \in C_k}\sum_{d_j \in C_{k'}}\frac{dis\left(d_i,d_j\right)}{|C_k|\,|C_{k'}|}\right]$$

Where d is the tweet represented as a vector, C is the cluster or topic, K is the number of topics. The model is considered high quality and comparable with human representation of topic clustering if it clusters and organizes the tweets into meaningful topics whereby tweets within a topic are coherent and substantively interrelated and simultaneously uncorrelated of other topics tweets. In order to calculate this, we measure the H score for each model. An ideal model has H score equal to zero. Therefore, the best model is the one corresponding to the lowest $Hscore$.

$$Hscore = \frac{Average\ \ Intra\ \ Dis}{Average\ \ Inter\ \ Dis}$$

We calculate the $Hscore$ for LDA, NMF and our "co-occurrence hashtags" model as results shown in Table 1 below.

Table 1: $HScore$ value for each model shows **co-occurrence hashtags** the lowest score indicating the better and more coherent topics

Model	**H Score**
co-occurrence hashtags	**0.353**
NMF	0.38
LDA	0.39

5.3 Human Judgment

Additionally, we perform human evaluation similar to the work proposed in [29]. Three annotators rate the quality of given topic's words semantic correspondence. They were asked to rate relevance by indicating on a scale from 1 (Very

Irrelevant) to 5 (Very Relevant). Then, we calculate the *Precision*, and record the results in a Table 2, for each method using the following equation:

$$Precision = \frac{Num\ of\ Relevant\ Topics}{Total\ Num\ of\ Topics}$$

Where the relevant topic is the topic with a score equal to or greater than 3 (out of 5). We measure the reliability of annotations using Fleiss' kappa metric [9], which measures the degree of agreement between multiple annotators. The resultant kappa factor is $k = 0.63$ which suggests a "substantial agreement" according to Landis et al. [16]. From Table 2, our two first models reach the highest consensus among annotators of topics quality leading to highest procession, yet the **co-occurrence hashtags** scores lower than **co-occurrence hashtags + NMF**. Such a lower score can be justified where some topical clusters show few atypical and other heterogeneous results which are not highly relevant. However, its overall outcome shows a far significant topics quality when compared to NMF or LDA alone.

Table 2: Precision results for each model and per the three annotators. This results suggests an average of co-occurrence hashtags (with and without the NMF) to be 0.85

Model	**Annotator 1**	**Annotator 2**	**Annotator 3**	**Average**
co-occurrence hashtags	**.95**	**1.**	**.95**	**.96**
co-occurrence hashtags + NMF	**.65**	**.7**	**.85**	**.73**
NMF	.4	.45	.8	.55
LDA	.05	.2	.75	.31

6 Conclusion and Future Work

We presented an approach in which we detect coherent topics on Twitter that assists in topic discovery. The goal is to automate and improve the quality and interpretability of discovered topics. Our work is language-independent, and it follows a technique that merges scattered and related hashtags representing topics through hashtag co-occurrence. Then, we identified the most representative hashtag as the topic's label. Since not all of the tweets have hashtag and in order to not lose a valuable information from these tweets, we utilize NMF to extract topics from the remaining data. The model was evaluated quantitatively and qualitatively. It achieved the highest topic coherence with $Hscore$ of 0.353 corresponding to the best cohesive outcome topics amongst others. Also, through

manual evaluation, it received the highest precision, with an average of 0.85 (out of 1.0). In our future work, we plan to enhance the framework by employing spell correction over tweets content and improving the expansion component of the framework pipeline for better topic detection and discovery.

References

1. Alsaedi, N., Burnap, P.: Arabic event detection in social media. In: International Conference on Intelligent Text Processing and Computational Linguistics. Springer (2015)
2. Alsaedi, N., Burnap, P., Rana, O.: Sensing real-world events using arabic twitter posts. In: Tenth International AAAI Conference on Web and Social Media (2016)
3. Alsaedi, N., Burnap, P., Rana, O.: Can we predict a riot? disruptive event detection using twitter. ACM Transactions on Internet Technology (TOIT) **17**(2), 18 (2017)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research (2003)
5. Boutsidis, C., Gallopoulos, E.: Svd based initialization: A head start for nonnegative matrix factorization. Pattern recognition (2008)
6. Chaffey, D.: Smart insights. 2019. Global Social Media Research Summary (2019)
7. Chen, Y., Zhang, H., Liu, R., Ye, Z., Lin, J.: Experimental explorations on short text topic mining between lda and nmf based schemes. Knowledge-Based Systems (2019)
8. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM (2006)
9. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological bulletin (1971)
10. Groß-Klußmann, A., König, S., Ebner, M.: Buzzwords build momentum: Global financial twitter sentiment and the aggregate stock market. Expert Systems with Applications (2019)
11. Hammad, M., El-Beltagy, S.R.: Towards efficient online topic detection through automated bursty feature detection from arabic twitter streams. Procedia Computer Science (2017)
12. Hansen, P.C.: The truncatedsvd as a method for regularization. BIT Numerical Mathematics (1987)
13. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. pp. 289–296. Morgan Kaufmann Publishers Inc. (1999)
14. Hong, L., Davison, B.D.: Empirical study of topic modeling in twitter. In: Proceedings of the first workshop on social media analytics. acm (2010)
15. Ilina, E., Hauff, C., Celik, I., Abel, F., Houben, G.J.: Social event detection on twitter. In: International Conference on Web Engineering. Springer (2012)
16. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. biometrics (1977)
17. Luo, J., Du, J., Tao, C., Xu, H., Zhang, Y.: Exploring temporal suicidal behavior patterns on social media: Insight from twitter analytics. Health informatics journal p. 1460458219832043 (2019)
18. Ma, Z., Sun, A., Cong, G.: On predicting the popularity of newly emerging hashtags in t witter. Journal of the American Society for Information Science and Technology pp. 1399–1410 (2013)

19. Marshakova, I.: Bibliographic coupling system based on references. Nauchno-Tekhnicheskaya Informatsiya Seriya, Ser **2**(6), 3–8 (1973)
20. Mimno, D., Wallach, H.M., Talley, E., Leenders, M., McCallum, A.: Optimizing semantic coherence in topic models. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics (2011)
21. Newman, D., Lau, J., Grieser, K., Baldwin, T.: Automatic evaluation of topic coherence. inhuman language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics, hlt'10 (2010)
22. Ozdikis, O., Senkul, P., Oguztuzun, H.: Semantic expansion of hashtags for enhanced event detection in twitter. In: Proceedings of the 1st international workshop on online social systems. Citeseer (2012)
23. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
24. Prier, K.W., Smith, M.S., Giraud-Carrier, C., Hanson, C.L.: Identifying health-related topics on twitter. In: International conference on social computing, behavioral-cultural modeling, and prediction. Springer (2011)
25. Rafea, A., Gaballah, N.A.: Trending topic extraction from twitter for arabic speaking user. In: The 33rd International Conference on Computers and Their Applications (CATA 2018), Las Vegas, Nevada, USA (2018)
26. Röder, M., Both, A., Hinneburg, A.: Exploring the space of topic coherence measures. In: Proceedings of the eighth ACM international conference on Web search and data mining. ACM (2015)
27. Roesslein, J.: Tweepy. Python programming language module (2015)
28. Stier, S., Bleier, A., Lietz, H., Strohmaier, M.: Election campaigning on social media: Politicians, audiences, and the mediation of political communication on facebook and twitter. Political communication (2018)
29. Wold, H.M., Vikre, L., Gulla, J.A., Özgöbek, Ö., Su, X.: Twitter topic modeling for breaking news detection. In: WEBIST (2) (2016)
30. Yan, X., Guo, J., Lan, Y., Cheng, X.: A biterm topic model for short texts. In: Proceedings of the 22nd international conference on World Wide Web. ACM (2013)
31. Zheng, C.T., Liu, C., San Wong, H.: Corpus-based topic diffusion for short text clustering. Neurocomputing (2018)

A Real Time Accident Detection Framework for Traffic Video Analysis

Hadi Ghahremannezhad, Hang Shi, and Chengjun Liu

New Jersey Institute of Technology
Department of Computer Science
Newark, NJ 07102 USA

Abstract. Traffic accident detection is an important topic in traffic video analysis, and this paper discusses single-vehicle traffic accident detection. Specifically, a novel real-time traffic accident detection framework, which consists of an automated traffic region detection method, a new traffic direction estimation method, and a first-order logic traffic accident detection method, is presented in this paper. First, the traffic region detection method applies the general flow of traffic to detect the location and boundaries of the roads. Second, the traffic direction estimation method estimates the moving direction of the traffic. The rationale for estimating the traffic direction is that the crashed vehicles often make rapid changes of directions. Third, traffic accidents are detected using the first-order logic decision-making system. Experimental results using the real traffic video data show the feasibility of the proposed method. In particular, traffic accidents are detected in real-time in the traffic videos without any false alarms.

Keywords: Accident detection, intelligent transportation, smart cities.

1 Introduction

Vehicle accidents on major roads and highways are one of the main issues in traffic management. It is important to report the accidents immediately when they occur, so that they can be dealt with without much delay. Automatic detection of traffic accidents helps turn traffic back to normal and if needed further medical assistance may be requested in a timely fashion. The term accident on the road may refer to different scenarios, such as rear-end, side-impact, head-on collisions, vehicle rollovers, or single-car accidents. The focus of this study is on single-vehicle accidents when a vehicle strikes a stationary object such as a tree or a barrier on the side of the road. Such incidents are usually caused by the driver losing control of the vehicle and making a sudden turn towards the road-side when there is no turning point.

In order to detect accidents on a highway involving vehicles, the first step is to detect and separate them from the background. Background subtraction methods based on the Gaussian mixture models are statistical techniques that provide a suitable approach to extract the foreground objects with a relatively low time complexity [3]. We apply the Global Foreground Modeling (GFM) method [14] for foreground detection. Note that the GFM method is chosen due to its robustness to noise, efficiency, and its ability to

keep the temporarily stopped objects in the foreground model. This is helpful in cases where the vehicles involved in an accident stop on the road after the accident.

After the moving objects are detected, they should be tracked as long as they are present in the scene in order to monitor their behavior and classify specific types of motion patterns. We apply the blob tracking method introduced in [4] for vehicle tracking. Note that this blob tracking method does not always track the vehicle continuously, but it is chosen for real-time vehicle tracking due to its simplicity and low computational complexity. Note also that in the process of accident detection the vehicle only needs to be tracked for a short period of time when it is involved in the accident.

The idea of our proposed real-time single-vehicle traffic accident detection framework analyzes the motion of each vehicle and applies heuristics to decide whether the pattern of movement matches those of single-vehicle accidents. First, the boundaries of the active traffic region are automatically detected using the foreground masks, which is explained in details in section 4.1. Second, the direction and the speed of a vehicle are examined as for a single-vehicle accident to take place, the vehicle should move towards the side of the road with rather high speed. The tracking information is utilized to estimate the direction for each vehicle, which is detailed in section 4.2. The average direction of the vehicles is calculated to estimate the correct moving direction at each part of the active traffic region. Finally, after noticing a vehicle making a sudden turn and moving outside of the traffic region, the variations in speed and neighboring foreground pixels are examined to decide whether a single-vehicle crash has happened. Section 4.3 explains the specific method.

This paper is organized as follows. In section 2, we will outline the previous related work that have approached the problem from various angles of view and mention their differences to our proposed method. In section 3 a statistical foreground modeling method from [14] that has been used to detect the moving and stopped vehicles is reviewed. In section 4 the three parts of the proposed method for traffic accident detection are described in detail. The performance of the proposed method is evaluated on different videos containing a single-vehicle accident in section 5, and we conclude the paper in section 6.

2 Related work

Over the past several decades there have been some studies addressing the issue of vision-based accident detection on roads and highways. Zu et al. [20] use a Gaussian Mixture Model to detect the moving vehicles and the mean shift method for tracking them. In this study three main motion features, namely, velocity, acceleration, and orientation are derived from the trajectories of the tracked vehicles. When all these values exceed the predefined thresholds, an accident is reported. Ren et al. [11] use a modified Gaussian Mixture Model to extract the moving vehicles in aerial videos and after detecting the lanes and dividing each lane into a cluster of cells, some traffic features are extracted for each cell based on the tracking information. Finally, a support vector machine is trained to detect incident points. Traffic parameters include flow rate, average travel speed, and average space occupancy. In [18] a close to real-time approach is proposed that divides each frame into non-overlapping blocks for each of which an average

velocity magnitude is calculated and the low-rank matrix approximation is utilized to detect the increase in approximation error. Although this method is more generalizable to different situations it can result in some false alarms. Also, the method can be computationally expensive for higher resolution videos. Maaloul et al. [8] use the Farneback optical flow to extract motion and a statistic heuristic approach to select thresholds and adaptively model traffic flow for accident detection.

Some other studies use more complex methods to detect abnormality in the traffic flow. In [16] vehicle incident analysis is formulated as an optimization problem. An optimal summarization framework is proposed which relies on the salient features of the moving vehicles. This method achieves comparatively good results. However, it suffers from errors in segmentation techniques. Ahmadi et al. [1] use a group sparse topical coding-based technique to model the normal traffic motion using the Lukas-Kanade's optical flow vectors in a document of words. In this model, each word corresponds to velocities in a specific range of orientations and when the computed words do not match the model it means some abnormal motion has happened. This approach is focused mostly on abnormal movement detection and is not specific to a type of accident. In [9] a three-stage approach is proposed to detect car crash incidents. First cars are detected using the You Only Look Once (YOLO) deep learning model. Then after tracking each detected car the Violent Flow (ViF) descriptor is used alongside an SVM to detect car crashes. This approach is not real-time, and there can be some false alarms. Xu et al. [19] present a model for anomaly detection in road traffic by analyzing vehicle motion patterns in static and dynamic modes. In the static mode, the background is subtracted and fed into a Faster R-CNN model for detecting stopped vehicles. In the dynamic mode, the trajectories of vehicles are tracked to find abnormal trajectory which is aberrant from the dominant motion patterns. This method ranked first place on the NVIDIA AI City Challenge. However, it has some limitations due to the use of a supervised deep learning model.

There have been more studies for vision-based traffic accident detection with the use of deep convolutional networks in recent years. In [2] Batanina et al. use a video game to generate synthetic data due to the lack of real videos of car crashes. After training a 3D deep convolutional neural network on the synthetic rendered videos domain adaptation is used to adapt the model to real videos. In [7] Huang et al. an integrated two-stream convolutional network architecture is proposed to detect and track the vehicles in real-time and also detect near-accidents in videos from overhead cameras. Appearance and motion features from the two networks are incorporated to detect near-accidents. Wang et al. [17] train a Yolo v3 model to detect objects in traffic videos that are related to different vehicle crash accidents, e.g., fallen pedestrians/cyclists, vehicle rollovers, and stopped vehicles. Then a decision model is trained based on a set of features developed from the outputs of the Yolo model to detect vehicle crash incidents.

Most of these studies are generally designed to detect abnormal traffic motion which can include stopped vehicles, head-to-head collisions, unexpected congestion, etc. and they are not specific to the type of anomaly. Some methods cannot be applied in real-time due to computational complexity. Also, many of the existing methods rely on supervised data to train a prediction model before they can be applied. In this paper, we present a novel real-time traffic accident detection framework to detect a specific type of road traffic accidents, namely, single-vehicle traffic accidents.

(a) Original frame (b) MOG foreground (c) GFM foreground

Fig. 1. The foreground masks extracted using the MOG method and the GFM method, respectively. Note that the GFM method extracts a more accurate foreground mask with both the moving vehicles (blue) and the stopped vehicles (red) clearly detected in the binary mask. In comparison, the MOG method fails to detect the stopped vehicles.

3 A Statistical Modeling Method for Detecting Both Foreground Objects and Stopped Moving Objects

Vehicle traffic accidents often involve moving vehicles and stopped moving vehicles, as when a traffic accident occurs, a vehicle is initially moving and then stops. Therefore traffic accident detection requires a method that is capable of detecting both foreground objects and stopped moving objects. We introduce in this section a statistical modeling approach that applies the Global Foreground Modeling (GFM) method [13], [14], the Mixture of Gaussian (MOG) method [15], and the Bayes Classifier to detect the foreground objects.

The GFM method models the foreground objects using a mixture of Gaussian distributions. Taking advantage of the fact that the foreground objects appear at different locations in some continuous frames, the GFM method models all the foreground pixels globally. In addition, the GFM method updates its parameters as the video progresses in order to adapt to different foreground objects. The global foreground model is described as follows:

$$P(\mathbf{x}|M_f) = \sum_{k=1}^{K} W_k N(\mathbf{x}|\omega_k) \tag{1}$$

$$N(\mathbf{x}|\omega_k) = \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^t \Sigma_k^{-1}(\mathbf{x}-\mu_k)\right\}}{(2\pi)^{d/2} \mid \Sigma_k \mid^{1/2}} \tag{2}$$

$$\sum_{k=1}^{K} W_k = 1 \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector that describes each pixel, M_f means the foreground class, K is the number of Gaussian distributions in the foreground model, W_k is the weight of the k_{th} Gaussian distribution $N(\mathbf{x}|\omega_k)$. μ_k and Σ_k are the mean vector and the covariance matrix of the k_{th} Gaussian density $N(\mathbf{x}|\omega_k)$. Note that every pixel that is classified as foreground is used to update the foreground model $P(\mathbf{x}|M_f)$. The foreground model is called global because it contains all the information of foreground pixels in the frame.

After the global foreground modeling, we also need to estimate a background model. We use the traditional MOG method, which estimates a Gaussian mixture density function for every location in a frame as the background model. The probability density function $P(\mathbf{x}|M_b, L)$ is calculated for location L as described in [15].

In order to classify a pixel into a foreground class or a background class, we apply the Bayes classifier for the classification.

$$p\left(\mathbf{x}|M_f, L\right) P(M_f, L) > p\left(\mathbf{x}|M_b, L\right) P\left(M_b, L\right) \tag{4}$$

For each pixel in a frame, if the inequality 4 holds, the pixel is classified as a foreground pixel. Otherwise, it is classified as a background pixel. Note that the conditional probability density functions $p(\mathbf{x}|M_f, L)$ and $p(\mathbf{x}|M_b, L)$ are estimated using the GFM model and the MOG model, respectively. The prior probabilities $P(M_f, L)$ and $P(M_b, L)$ are estimated using the weights of the MOG model [6].

Fig. 1 shows the foreground masks extracted using the MOG method and the GFM method, respectively. Note that the GFM method extracts a more accurate foreground mask with both the moving vehicles and the stopped vehicles clearly detected in the binary mask. In comparison, the MOG method fails to detect the stopped vehicles.

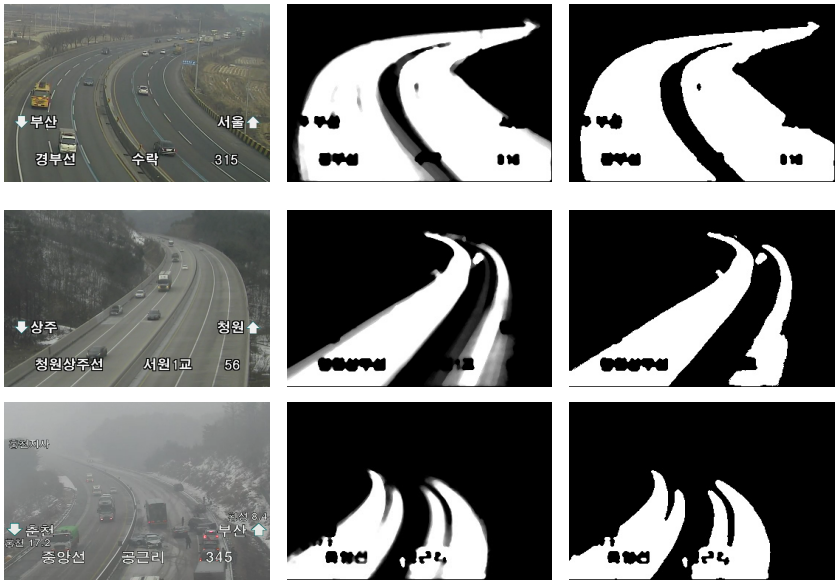
4 A Novel Real Time Traffic Accident Detection Framework

Our proposed real-time single-vehicle traffic accident detection framework consists of three major methods: an automated traffic region detection method, a new traffic direction estimation method, and a traffic accident detection method using the first-order logic. These three methods detect the active traffic region, estimate the traffic direction, and detect the single-vehicle traffic accidents by applying the assumptions about the abnormality of the movement and specific behaviors of a vehicle that lead to crashing to the traffic barrier.

4.1 An Automated Traffic Region Detection Method

We can use the general flow of traffic to obtain some information about the location and boundaries of the roads. When a vehicle is moving along the road it usually follows the right direction within the predefined lanes. The foreground mask from the GFM method contains a white blob for each vehicle. Every time a vehicle passes the road, we add a portion of its mask color to the estimated road. After enough vehicles have passed the road these color values increase to the point that the road section is estimated with a high degree of certainty. By applying the Otsu's threshold [10, 5] we can get rid of noise and the areas outside of the road, that are usually added by the mask of larger vehicles like trucks or buses, and obtain a binary image representing the estimated zone of the traffic flow. The boundaries of the active traffic region becomes closer to the road boundaries as more vehicles pass along the road.

$$r_c = \sum_{f=1}^{T} m_f / F \tag{5}$$



(a) (b) (c)

Fig. 2. Estimation of the active traffic area using the cumulative foreground mask of the GFM method and the Otsu's threshold. (a) The original frame from traffic video. (b) Accumulative foreground mask of the moving vehicles. (c) Estimated traffic region using Otsu's method.

$$r_f = \begin{cases} 1, & \text{if } r_c \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where r_f is the final traffic region binary map, r_c is the accumulative traffic region map, f is the current frame, τ is the calculated Otsu's threshold, m_f is the foreground mask of frame f, and F is the number of frames.

Some traffic videos are captured by cameras overlooking the highway from one side of the road. In this type of videos, the boundary between two directions may not be accurate since the foreground masks of vehicles in two opposite directions overlap. The foreground mask of vehicles that are closer to the camera can cover the ones of the further side. In such cases, this boundary can be detected by considering the direction of movement of each vehicle to separate the foreground masks. Fig. 2 shows some samples of the traffic area estimation using the cumulative foreground mask of the GFM method and the Otsu's threshold. In the case of videos with shorter lengths, the number of passing vehicles might not be enough to cover the entire surface of the road sections. Also, the stationary captions on the videos block parts of the scene and are always classified as part of the background. Therefore, some gaps can be seen in the map of the traffic region, some of which can be covered as time passes and more vehicles move along the road.

4.2 A New Traffic Direction Estimation Method

The first step after segmenting the foreground and tracking the vehicles is to estimate the traffic direction on the road. Since in many traffic videos the roads are curved we cannot use a single direction for the entire road segment. Therefore, we divide the road into a number of rectangular areas and estimate one traffic direction for each of these areas based on the average direction and magnitude of the moving vehicles for each area of the road. A number of frames (f) are used to estimate the direction of each vehicle. This is done by finding the mean centroid from the first half and the second half of the f frames to estimate a consistent and smooth direction for the movement of each vehicle. The direction and magnitude of each vehicle are estimated as follows:

$$\begin{aligned} v_x &= x_{m_2} - x_{m_1} \\ v_y &= y_{m_2} - y_{m_1} \\ d_i &= arctan(v_y, v_x) \\ m_{v_i} &= \sqrt{v_x^2 + v_y^2} \end{aligned} \tag{7}$$

where v_x and v_y are the components of the velocity vector, x_{m_2} and y_{m_2} are the mean x and y values of the blob centroid in the most recent $f/2$ frames, x_{m_1} and y_{m_1} are the mean x and y values of the blob centroid in the remaining $f/2$ frames, d_i is the estimated direction of the vehicle i, and m_{v_i} is the estimated magnitude of the vehicle i, respectively. Note that we do not consider the slow movements for the direction estimation; since in these cases, the centroids are too close which can lead to faulty results. Furthermore, the vehicles have to be mostly separated and in situations when traffic congestion occurs, average directions are not updated. Therefore, only movements with considerable speed and size are considered for estimating the average direction and the average speed.

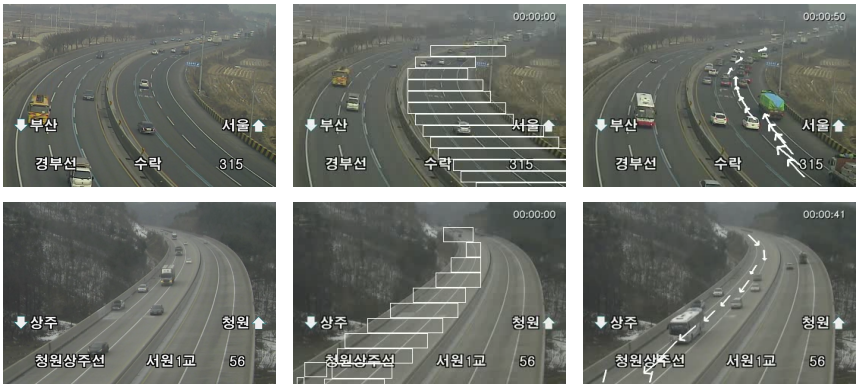
After the calculation of the moving direction of the vehicles the average direction and magnitude at each part of the active traffic region can be estimated based on equations 8 and 9 for each frame.

$$avg(c_{k_d}) = (1/F) * \sum_{f=1}^{F} \frac{\sum_{i=1}^{n} d_{i,f}}{N} \tag{8}$$

$$avg(c_{k_m}) = (1/F) * \sum_{f=1}^{F} \frac{\sum_{i=1}^{n} m_{i,f}}{N} \tag{9}$$

where c_{k_d} and c_{k_m} are the direction and magnitude of $cell_k$ respectively, F is the total number of frames, f is the current frame, n is the number of vehicles at $cell_k$, N is the total number of vehicles passed through $cell_k$, and $d_{i,f}$ and $m_{i,f}$ are the direction and magnitude of vehicle i at frame f respectively, which are calculated based on equation 7.

Fig. 3 shows the estimation of the traffic flow direction at each area of the curved road. The size of these areas can be estimated by considering the size of the road section and the average size of the vehicles. To partition the road we used the contour derived



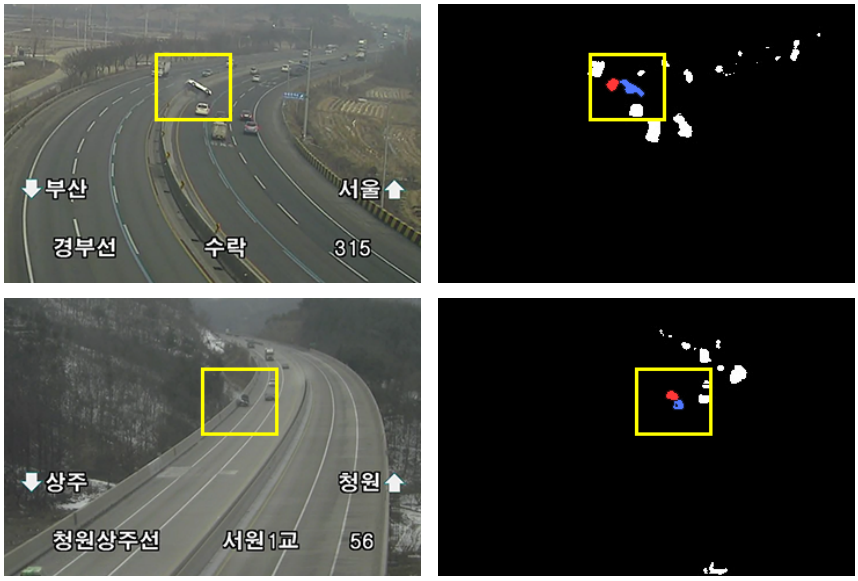
(a) (b) (c)

Fig. 3. Estimation of the traffic flow direction at each area of the curved road. (a) The original frame from a traffic video. (b) The automatically partitioned rectangular areas of the curved road. (c) The estimated average direction and magnitude of the moving vehicles for each part of the road.

from the estimated traffic region map. Multiple squares are drawn over the contour where the area of the contour in each square exceeds a threshold percentage and finally, the squares at the same row are joined together to form a rectangle.

When a vehicle hits the traffic barrier it usually starts with an abrupt movement which is mostly caused by the driver losing control of the vehicle. This rapid movement can be detected by comparing the direction of the moving vehicle and the estimated direction for the area of the road where the vehicle is currently on. If the two degrees differ more than a notable value (d) and the magnitude of the movement is also large, it means that the vehicle is making a sudden unexpected move that often can be dangerous. This kind of hasty movement alone does not necessarily result in the vehicle colliding the traffic barrier or another vehicle.

We should also consider the location of the vehicle after it has made a hasty move. If the vehicle goes out of the estimated boundary of the road without slowing down its speed, there can be two possibilities. Either the vehicle is making a turn to another road that is not detected in road estimation (because there have not been enough cars making a similar turn), or the driver is making a rapid side move which can be due to losing control. For the first case, the vehicle will not crash and will continue its movement and that road will be added to the estimated road map. However, if the vehicle actually hits an obstacle it will most probably have a considerable change in its speed, direction, and acceleration. In some scenarios, this type of accident may also lead to vehicle rollovers. After the traffic accident, the vehicle itself and some of the surrounding vehicles usually stop and traffic congestion occurs. All these cues can help detect a single-vehicle traffic accident. Furthermore, another cue of a single-vehicle collision can be the foreground segmentation mask showing a splash (an unexpected blob detected in the middle of the road) caused by the vehicle hitting the traffic barrier (see Fig.4).



(a) (b)

Fig. 4. Unexpected blob detected in the middle of the foreground mask caused by the vehicle hitting the traffic barrier. The vehicle and the unexpected emerged blob are indicated by blue and red colors respectively. (a) The original frame from traffic video. (b) The foreground mask and the unexpected blob caused by the vehicle crashing the traffic barrier.

4.3 A Traffic Accident Detection Method Using The First-Order Logic

By considering the occurrence of a sequence of steps we are able to detect single-vehicle collisions. To keep track of the target vehicle we can use the stopped vehicle strategy as a factor that makes the assumption more certain. To detect whether the vehicle is stopped, we use the foreground mask from the GFM method which keeps the corresponding foreground information for temporarily stationary objects. Due to the fact that in most cases the vehicle stops after having an accident and there might be some level of congestion and slow traffic flow. In other words, the probability of accident having taken place is high if the same vehicle stops after the abnormal movement and if the nearby vehicles also stop or move at a slow speed. We can make an assumption about an accident occurring after having all these incidents happened in close proximity to each other. Here we consider all these factors in order to decide on the possibility of a single-vehicle traffic accident.

The first step of the proposed method is to estimate the location and boundaries of the two directions of the road by thresholding their accumulative foreground masks. As the number of vehicles passing through different parts of the road grows, the probability of that region belonging to the road increases. This step is useful for having an estimation of the correct traffic zone and the boundaries of the road.

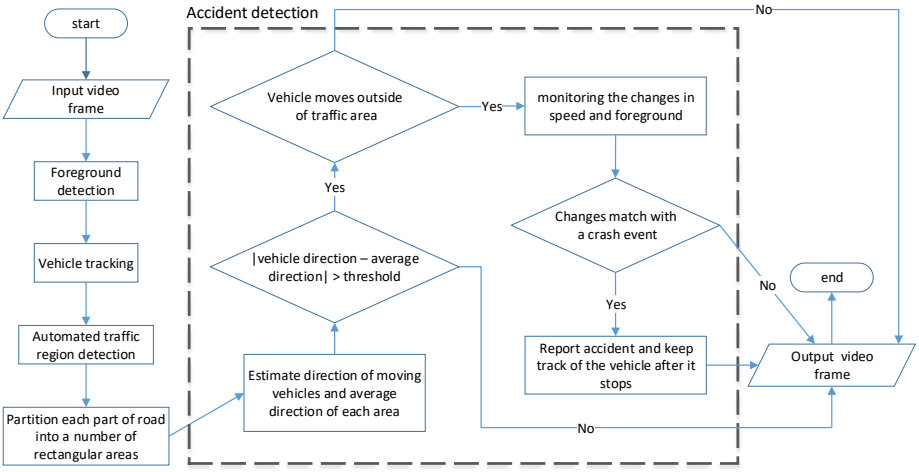


Fig. 5. Flowchart of the proposed real time single-vehicle traffic accident detection framework.

The second step is to partition each part of the road into a number of rectangular areas, each of which has an average direction, average speed, and average blob size. The purpose of dividing the road into different areas is to estimate the direction of the traffic flow at each area of the road. Note that while on straight roads the direction of the traffic flow does not change much, but on curved roads, the direction changes rapidly. Therefore, partitioning the traffic region into smaller areas and assigning a unique average direction and speed to each of them can help improve traffic accident detection accuracy. The rectangular areas on each side of the road are calculated automatically based on the contour of the active traffic region map for that side. Each rectangular area covers the width of the road at the corresponding location and the height of each rectangle is set to be small enough to be reliable even at the curvy parts of the road.

The third step of the proposed method is to detect in real-time single-vehicle traffic accidents. Since crashing the barrier usually starts with an abrupt side-move, the direction of each tracked vehicle (not considering slow vehicles) is compared with the average direction of the corresponding area (part of the road where the vehicle is currently on). If a vehicle makes a rapid side-move, we keep track of that vehicle to see whether it moves out of the road boundaries or the abrupt movement ends earlier. In case the vehicle moves out of the traffic region the changes in speed and neighboring foreground mask are monitored. If the speed decreases suddenly and an unexpected foreground blob appears in the proximity of the vehicle, it indicates that a crash has happened. Fig. 5 shows the flowchart of the proposed real-time single-vehicle traffic accident detection framework.

The idea of our proposed real-time single-vehicle traffic accident detection framework may be expressed using the first-order logic knowledge representation language [12]. In particular, the following statements 10, 11 and 12 represent the idea for traffic accident detection.

$$\forall v Vehicle(v) \wedge Fast(v) \wedge Swerve(v) \\ \wedge \neg ShortDistance(v) \Rightarrow Rapid(v) \tag{10}$$

where v represents a vehicle, $Vehicle(v)$ means that v is an actual tracked vehicle that is in the current frame, $Fast(v)$ means that the estimated magnitude for v is around the average magnitude of the cell containing its centroid or higher, $Swerve(v)$ indicates that the calculated direction of movement for vehicle v is different from the average direction of the cell containing its centroid by a value more than $45°$. $ShortDistance(v)$ stipulates that the size of movement should not be too small in order to avoid false positives caused by the inaccuracies in the blob detection process. $Rapid(v)$ means that vehicle v has made an abrupt side-move in an unexpected location (see Fig. 7(b)). These types of movements do not always result in the vehicle crashing an obstacle on the side of the road. Therefore, in order to draw a conclusion that an accident has happened more information needs to be considered.

$$\forall v Vehicle(v) \wedge OutOfBoundary(v) \\ \wedge Splash(v) \Rightarrow Crash(v) \tag{11}$$

where $OutOfBoundary(v)$ is a predicate which indicates that vehicle v has moved outside of the estimated traffic region, $Splash(v)$ means that there is an unexpected blob in the foreground mask in the surrounding block of vehicle v, and $crash(v)$ means that vehicle v has probably collided with some obstacle on the side of the road.

$$\forall v Vehicle(v) \wedge Rapid(v) \wedge Crash(v) \\ \wedge TimeOf(Rapid(v)) < TimeOf(Crash(v)) \\ \Rightarrow Accident(v) \tag{12}$$

where $TimeOf()$ is a function which returns the time when its input term has occurred, and $Accident(v)$ indicates that vehicle v has had a single-vehicle accident. Therefore, this statement means a single-vehicle crash happens when a vehicle hits the barrier after moving to that direction with a high speed without slowing down during this abrupt movement.

To prove the rules are complete for FOL, we can use the forward chaining method which is complete for a Horn knowledge base. The knowledge base is a set of information representing facts about a particular subject. As for these facts in case of a single-vehicle road-side accident we have assumed if a vehicle makes a sudden turn to the side with high enough speed and long enough moving distance, it has made a dangerous move which we call a rapid move. Also, we assume if a vehicle moves outside of the common traffic region boundaries and at the same time a blob of pixels appear in the foreground mask around the vehicle a collision with an obstacle might have happened which we call crash.

For a single-vehicle road-side accident to occur, the rapid move should happen before the crash. If we consider V_1 to be a vehicle experiencing both incidents in chronological order the occurrence of a single-vehicle accident can be concluded. To use the

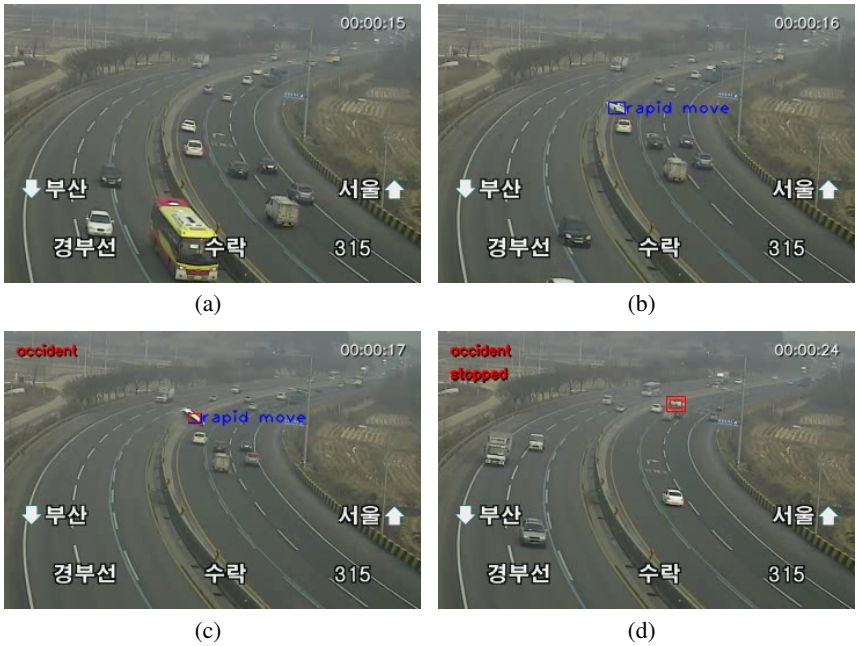


Fig. 6. Real time single-vehicle traffic accident detection results using a real traffic video. (a) Vehicles move in the correct traffic direction. (b) A vehicle makes a sudden side move. (c) The vehicle hits the road barrier. (d) The vehicle stops after the accident.

forward chaining method we can use the known facts to keep proving new information and eventually prove the final clause. Assuming a vehicle V_1 has met all the preconditions of a single-vehicle accident we can use the known facts to prove the accident has occurred.

According to the statement 10 which is in the form of a Horn clause the rapid movement of the vehicle V_1 can be proved by considering four facts from knowledge-base to be true for this vehicle. These facts being that V_1 is a vehicle and it has made a large movement with a high speed. In line with statement 11 which is also in the form of a Horn clause the predefined crash incident can be concluded by considering two more facts from knowledge-base to be true about vehicle V_1 that are moving outside of the traffic region boundaries and occurrence of an unexpected foreground blob around the vehicle. As stated in 12, the resulting clause which is $Accident(V_1)$ can then be concluded by conjunction of the previous Horn clauses with another fact from knowledge based that indicated the right time order.

5 Experiments

We apply real traffic videos from the department of transportation to evaluate our proposed method. The spatial resolution of the traffic videos used in our experiments is

Fig. 7. Real time single-vehicle traffic accident detection results using a real traffic video. (a) Vehicles moves in the right traffic direction. (b) A vehicle makes a sudden side move. (c) The vehicle hits the road barrier. (d) The vehicle stops after the accident.

720×480 with a frame-rate of 30 frames per second. Specifically, first, the motion information from the videos is used to estimate the road boundaries. Second, the tracking and the foreground segmentation results are applied to detect the abnormal motion. And finally, the first-order logic decision-making system is utilized to detect single-vehicle accidents. The traffic accidents are detected in real-time in the traffic videos without any false alarms. The experiments are implemented using a DELL XPS 8900 PC with a 3.4 GHz processor and 16 GB RAM.

Fig. 6 and Fig. 7 show the experimental results of real-time single-vehicle traffic accident detection using two real traffic videos from the department of transportation. Considering the limitation of video data for the specific type of single-car traffic accidents we only apply our method on two video sequences. In particular, Fig. 6 (a) shows that the vehicles are moving in the right traffic direction in a frame from one traffic video. Fig. 6 (b) shows that a vehicle makes a sudden side move, which is detected automatically by our proposed method. Fig. 6 (c) shows that the vehicle hits the road barrier, and our proposed method automatically detects such a single-vehicle traffic accident in real-time. Fig. 6 (d) shows that the vehicle stops after the accident, and our proposed method automatically detects both the traffic accident and the stopped vehicle in real-time. Fig. 7 shows the real-time traffic accident results using another real traffic video from the department of transportation. Our proposed method successfully detects the vehicle's sudden move to the side of the road, the traffic accident when the vehicle

Table 1. Running time of the proposed method

	video 1	video 2
Length of video (s)	56	56
Running time (s)	44.70	42.26
Number of frames	1680	1680
Running time for each frame (ms)	26	25

hits the road barrier, as well as the stopped vehicles in real-time as shown in Fig. 7 (b), Fig. 7 (c), and Fig. 7 (d), respectively.

Table 1 shows the length (in seconds) of videos, the running time (in seconds) of our proposed method, the number of frames in each of the videos, and the running time (in milliseconds) for each frame. From the table, we can see that our proposed method runs in real-time.

6 Conclusion

We have presented in this paper a novel real-time single-vehicle accident detection method for traffic video analysis. First, we use a statistical foreground modeling method to detect the foreground objects. In order to detect both the moving foreground objects and the temporarily stopped objects, the Global Foreground Modeling (GFM) method is used together with the Mixture of Gaussian (MOG) method. In addition, the Bayes classifier is applied for foreground and background classification. Second, we propose our novel traffic accident detection method. The contributions of our proposed method are three-fold: (i) a new traffic region detection method, (ii) a traffic direction estimation method, and (iii) a single-car run-off-road accident detection method using the first-order logic. The traffic region detection method is used to find out the boundaries of the road. By detecting the road boundaries, we are able to detect the vehicles that hit or go outside the boundaries. The traffic direction estimation method is able to estimate the correct direction of the moving traffic. A vehicle with an abnormal moving direction may lead to a traffic accident. These two methods can provide some clues for detecting a traffic accident. Finally, we use the first-order logic to make a final decision based on these clues. We implement our proposed method and evaluate it using real traffic video data and achieve good performance in real-time traffic accident detection.

Acknowledgments: This work is partially supported by the NSF grant 1647170.

References

1. Ahmadi, P.: Abnormal event detection and localisation in traffic videos based on group sparse topical coding. IET Image Processing **10**, 235–246(11) (March 2016)
2. Batanina, E., Eddine, I., Bekkouch, I.B.I., Khan, A., Khattak, A.M., Bortnikov, M.: Domain adaptation for car accident detection in videos. In: 2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA). pp. 1–6. IEEE (2019)

3. Bouwmans, T., Garcia-Garcia, B.: Background subtraction in real applications: Challenges, current models and future directions. arXiv preprint arXiv:1901.03577 (2019)
4. Chang, F., Chen, C.J., Lu, C.J.: A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding **93**(2), 206–220 (2004)
5. Goh, T.Y., Basah, S.N., Yazid, H., Safar, M.J.A., Saad, F.S.A.: Performance analysis of image thresholding: Otsu technique. Measurement **114**, 298–307 (2018)
6. Hayman, E., Eklundh, J.: Statistical background subtraction for a mobile observer. In: 9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France. pp. 67–74. IEEE Computer Society (2003)
7. Huang, X., He, P., Rangarajan, A., Ranka, S.: Intelligent intersection: Two-stream convolutional networks for real-time near-accident detection in traffic video. ACM Transactions on Spatial Algorithms and Systems (TSAS) **6**(2), 1–28 (2020)
8. Maaloul, B., Taleb-Ahmed, A., Niar, S., Harb, N., Valderrama, C.: Adaptive video-based algorithm for accident detection on highways. In: 2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES). pp. 1–6 (June 2017). https://doi.org/10.1109/SIES.2017.7993382
9. Machaca Arceda, V.E., Laura Riveros, E.: Fast car crash detection in video. In: 2018 XLIV Latin American Computer Conference (CLEI). pp. 632–637 (Oct 2018). https://doi.org/10.1109/CLEI.2018.00081
10. Otsu, N.: A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics **9**(1), 62–66 (1979)
11. Ren, J.: Detecting and positioning of traffic incidents via video-based analysis of traffic states in a road segment. IET Intelligent Transport Systems **10**, 428–437(9) (August 2016), https://digital-library.theiet.org/content/journals/10.1049/iet-its.2015.0022
12. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, (2016)
13. Shi, H., Liu, C.: A new foreground segmentation method for video analysis in different color spaces. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 2899–2904. IEEE (2018)
14. Shi, H., Liu, C.: A new global foreground modeling and local background modeling method for video analysis. In: International Conference on Machine Learning and Data Mining in Pattern Recognition. pp. 49–63. Springer (2018)
15. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). p. 2246. IEEE (1999)
16. Thomas, S.S., Gupta, S., Subramanian, V.K.: Event detection on roads using perceptual video summarization. IEEE Transactions on Intelligent Transportation Systems **19**(9), 2944–2954 (Sep 2018). https://doi.org/10.1109/TITS.2017.2769719
17. Wang, C., Dai, Y., Zhou, W., Geng, Y.: A vision-based video crash detection framework for mixed traffic flow environment considering low-visibility condition. Journal of advanced transportation **2020** (2020)
18. Xia, S., Xiong, J., Liu, Y., Li, G.: Vision-based traffic accident detection using matrix approximation. In: 2015 10th Asian Control Conference (ASCC). pp. 1–5 (May 2015). https://doi.org/10.1109/ASCC.2015.7244586
19. Xu, Y., Ouyang, X., Cheng, Y., Yu, S., Xiong, L., Ng, C.C., Pranata, S., Shen, S., Xing, J.: Dual-mode vehicle motion pattern learning for high performance road traffic anomaly detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2018)
20. Zu, H., Xie, Y., Ma, L., Fu, J.: Vision-based real-time traffic accident detection. In: Proceeding of the 11th World Congress on Intelligent Control and Automation. pp. 1035–1038 (June 2014). https://doi.org/10.1109/WCICA.2014.7052859

Similarity Learning by Random Projection Forests with Application to Clustering

Donghui Yan[1], Songxiang Gu[2], Ying Xu[3], and Zhiwei (Tony) Qin[4]

[1] University of Massachusetts, Dartmouth, MA 02747, USA
[2] JD Digital, Mountain view, CA 94085, USA
[3] Indigo Agriculture Inc, Boston, MA, USA
[4] DiDi Research America, Mountain View, CA 94043, USA
dyan@umassd.edu, songxiang.gu@gmail.com, yingxuuc@gmail.com, qinzhiwei@didichuxing.com

Abstract. Similarity plays a fundamental role in many areas, including data mining, machine learning, statistics and various applied domains. Inspired by the success of ensemble methods and the flexibility of trees, we propose to learn a similarity kernel called *rpf-kernel* through random projection forests (*rpForests*). The rpf-kernel has a highly desirable property: *far-away (dissimilar) points have a low similarity value while nearby (similar) points would have a high similarity*, and the similarities have a native interpretation as the probability of points staying in the same leaf nodes during the growth of *rpForests*. The learned rpf-kernel leads to an effective clustering algorithm—*rpfCluster*. On a wide variety of real and benchmark datasets, *rpfCluster* compares favorably to K-means clustering, spectral clustering and a state-of-the-art clustering ensemble algorithm—Cluster Forests. Our approach is simple to implement and readily adapt to the geometry of the underlying data. Given its desirable theoretical property and competitive empirical performance when applied to clustering, we expect rpf-kernel to be applicable to many problems of an unsupervised nature or as a regularizer in some supervised or weakly supervised settings.
Index terms— Similarity learning, distance metric learning, clustering, random projection forests.

1 Introduction

Similarity measures how similar or closely related different objects are. It is a fundamental quantity that is relevant whenever a distance metric or a measure of relatedness is used. Our interest is when the similarity is explicitly used. A wide range of applications use the notion of similarity. In data mining, the answer to a database (or web) query is often formulated as similarity search [22,25] where instances similar to the one in the query are returned. More general is the k-nearest neighbor search, which has applications in robotic route planning [31], face-based surveillance systems [44], anomaly detection [2,13] etc. In machine learning, the entire class of kernel methods [47,28] rely on similarity or the similarity kernel. The similarity kernel is also an essential part of popular classifiers

such as support vector machines [16], the broad class of spectral clustering algorithms [50,43,53], and various kernalized methods such as kernel PCA [46], kernel ICA [3], kernel k-means [20] etc. Another important class of methods that use similarity is clustering ensemble [52,58] which use the similarity to encode how different data points are related to each other in terms of clustering when viewed from many different clustering instances. Additionally, the similarity is used as a regularization term that incorporates some latent structures of the data such as the *cluster* [14] in semi-supervised learning, or as regularizing features that capture the locality of the data (i.e., which of the data points look similar) [61]. In statistics, the notion of similarity is also frequently used (a significant overlap with those used in machine learning). It has been used as a distance metric in situations where the Euclidean distance is no longer appropriate or the distance needs to be adaptive to the data, for example in nonparametric density estimation [7,40] and intrinsic dimension estimation [8,33]. Another use is to measure the relatedness between two random objects, such as the covariance matrix. It is also used as part of the aggregation engine to combine data from multiple sources [54] or multiple views [21,57] etc.

The benefit of adopting a similarity kernel is immediate. It encodes important property about the data, and allows a unified treatment of many seemingly different methods. Also one may take advantage of the powerful kernel methods. However, in many situations, one has to choose a kernel that is suitable for a particular application. It will be highly desirable to be able to automatically choose or learn a kernel that would work for the underlying data. In this paper, we explore a data-driven approach to learn a similarity kernel. The similarity kernel will be learned through a versatile tool that was recently developed—random projection forests (*rpForests*) [62].

rpForests is an ensemble of random projection trees (rpTrees) [17] with the possibility of projections selection during tree growth [62]. rpTrees is a randomized version of the popular kd-tree [6,25], which, instead of splitting the nodes along coordinate-aligning axes, recursively splits the tree along randomly chosen directions. *rpForests* combines the power of ensemble methods [12,24,58] and the flexibility of trees. As *rpForests* uses randomized trees as its building block, accordingly it has several desired properties of trees. Trees are invariant with respect to monotonic transformations of the data. Trees-based methods are very efficient with a log-linear (i.e., $O(n\log(n))$) average computational complexity for growth and $O(\log(n))$ for search where n is the number of data points. As trees are essentially recursive space partitioning methods [6,17], data points falling in the same tree leaf node would be close to each other or "similar". This property is often leveraged for large scale computation [18,37,59,62] etc. Now it forms the basis of our construction of the similarity kernel—data points in the same leaf node are likely to be similar and dissimilar otherwise. Additionally, as individual trees are rpTrees thus can adapt to the geometry of the data and readily overcomes the curse of dimensionality [17].

Ideally, for a similarity kernel, similar objects should have a high similarity value while low similarity value for dissimilar objects. Similarity as produced by a

single tree may suffer from the undesirable boundary effect—similar data points may be separated into different leaf nodes during the growth of a tree. This will cause problem for a similarity kernel for which the similarity of every pair (or most pairs) of points matters. The ensemble nature of *rpForests* allows us to effectively overcome the boundary effect—by ensemble, data points separated in one tree may meet in another; indeed *rpForests* reduces the chance of separating nearby points exponentially fast [62]. Meanwhile, dissimilar or far-away points would unlikely end up in the same leaf node. This is because, roughly, the diameter of tree nodes keeps on shrinking during the tree growth, and eventually those dissimilar points would be separated if they are far away enough. Thus a similarity kernel produced by *rpForests* would possess the expected property.

Our main contributions are as follows. First, we propose a data-driven approach to learn a similarity kernel from the data by *rpForests*. It combines the power of ensemble and the flexibility of trees; the method is simple to implement, and readily adapt to the geometry of the data. As an ensemble method, easily it can run on clustered or multi-core computers. Second, we develop a theory on the property of the learned similarity kernel: similar objects would have high similarity value while low similarity value for dissimilar objects; the similarity values have a native interpretation as the probability of points staying in the same tree leaf node during the growth of *rpForests*. With the similarity kernel learned by *rpForests*, we develop a highly competitive clustering algorithm, *rpfCluster*, which compares favorably to spectral clustering and a state-of-the-art ensemble clustering method.

The remainder of this paper is organized as follows. In Section 3, we give a detailed description of how to produce a similarity kernel by *rpForests* and a clustering method based on the resulting similarity kernel. This is followed by some theoretical results on the similarity kernel learned by *rpForests* in Section 4. Related work are discussed in Section 2. In Section 5, we first provide examples to illustrate the desired property of the similarity kernel and its relevance in clustering, and then present experimental results on a wide variety of real datasets for *rpfCluster* and its competitors. Finally, we conclude in Section 6.

2 Related work

Similarity plays a very important role in machine learning. Due to the intimate connection between similarity and metric learning, we do not distinguish the two in this section. A simple similarity measure is typically defined through a closed-form function such as the cosine (or weighted cosine along principal directions [49]), Euclidean (or Minkoski) distance function etc. More sophisticated is the bilinear similarity which measures the similarity of any two objects $x_1, x_2 \in \mathbb{R}^p$ by a bilinear form

$$f_S(x_1, x_2) = (x_1 - x_2)^T S(x_1 - x_2)$$

where S is a symmetric positive semidefinite matrix. This allows to incorporate feature weights or feature correlations into the similarity, and the Mahalanobis distance [1] is a classic example. Indeed many research on similarity

learning starts from an early work on learning the *Mahalanobis distance* with side information such as examples of similar pairs of objects [56]. There are many followup work along this line, for example, [48] regularizes the learning by large margin, [4] uses side-information in the form of equivalence constraints, [19] minimizes the relative entropy between two Gaussians under constraints on the distance function, [55] learns a large-margin nearest neighbor metric such that k-nearest neighbors are of the same class while different classes otherwise. Later work either adds more constraints (such as sparsity) or extends the Mahalanobis metric, or on broader classes of problems (such as ranking); this includes [15,27,35,38,36,29]. Similarity or metric learning is a big topic, and it is beyond our scope to have a more thorough review on existing work; readers can refer to surveys [63,9,10,32,5,42] and references therein. Existing work are almost exclusively supervised or weakly supervised in nature, our work is different in that it is unsupervised.

Another related topic is clustering ensemble [52,26,58]. Clustering ensemble works by first generating many clustering instances, and then produce the final cluster by aggregating results from individual clustering instances. Literature on clustering ensemble is huge; readers can refer to [52,23,58,11] and references therein. Clustering ensemble is related as each tree in *rpfCluster* can be viewed as an instance of clustering where points in the same leaf node form a cluster. Most closely related are random projection based clustering ensemble [23] and Cluster Forests (CF) [58]. [23] generates individual clustering instances by random projection of the original data onto some low dimensional space and then perform clustering. *rpfCluster* differs by generating a clustering instance through the growth of a tree which iteratively refines the clustering, or, *rpfCluster* projects the data onto low dimensional spaces by a series of random projections with each to a one-dimension space. Same as *rpfCluster*, CF also iteratively improves clustering instances, and produces the final cluster by spectral clustering on the learned similarity kernel; the difference is that CF generates clustering instances by a base clustering algorithm and refines each clustering instances by randomized feature pursuits.

Finally there are connections to Random Forests (RF) [12]. Both RF and *rpfCluster* generates ensemble of trees with random ingredients; the difference is that RF is *supervised* as tree growth is guided by class labels while *rpfCluster* grows trees *unsupervisedly*, also RF splits on coordinates while *rpfCluster* on random projections. Both *rpfCluster* and CF can be viewed as *unsupervised extensions* to RF. CF aims at clustering by ensemble of iteratively refined clustering instances through randomized feature pursuits while *rpfCluster* learns the similarity kernel by ensemble of random projection trees. Another connection is that RF can run in unsupervised mode [12,51] to learn a suitable distance metric for clustering; this is done by synthesizing a contrast pattern through randomization of the original data by randomly permuting the data along each of its features thus breaking the covariance structure of the data (implemented by the *proximity* option in the R package "randomForest"), which is fundamentally different from our approach.

3 Proposed approach

In this section, we will first describe *rpForests*, and then discuss how to generate the similarity kernel with *rpForests* and to cluster with the similarity kernel, followed by a brief introduction to spectral clustering.

3.1 Algorithmic description of *rpForests*

Our description of the *rpForests* algorithm is based on [62]. Each tree in *rpForests* is an rpTree. The growth of an rpTree follows a recursive procedure. It starts by treating the entire data as the root node and then split it into two child nodes according to a randomized splitting rule. On each child node, the same splitting procedure applies recursively until some stopping criterion is met, e.g., the node becomes too small (i.e., contains too few data points).

In rpTree, the split of a node is along a randomly generated direction, say $\vec{r}$. Assume the node to split is W. Node W is split along its projection onto $\vec{r}$ according to a split point, say c, sampled uniformly at random over the interval formed by the projection of all points in W onto $\vec{r}$, denoted by $W_{\vec{r}} = \{P_{\vec{r}}(x) = r \cdot x : \ x \in W\}$. Let $V = \{X_1, ..., X_n\}$ denote the given data set. Let $\mathcal{W}$ denote the set of nodes to be split (termed as the *working set*). Let n_s denote a constant such that a node will not be split further if its size is smaller than n_s. Denote the *rpForests* by $\mathcal{F}$; assume there are totally T trees. The algorithm for generating *rpForests* is described as Algorithm 1.

Algorithm 1 *rpForests(V, T)*

Initialize $\mathcal{F} \leftarrow \emptyset$;
for $i = 1$ to T **do**
  Let V be the root node of tree t_i;
  Initialize the working set $\mathcal{W} \leftarrow \{V\}$;
  while $\mathcal{W}$ is not empty **do**
    Sample $W \in \mathcal{W}$ and update $\mathcal{W} \leftarrow \mathcal{W} \setminus \{W\}$;
    if $|W| < n_s$ **then**
      Skip to the next round of the loop;
    end if
    Generate a random direction $\vec{r}$ and project W onto $\vec{r}$;
    Sample splitting point c uniformly from the range spanned by $W_{\vec{r}}$;
    Split node W into $W_L = \{x : P_{\vec{r}}(x) < c\}$ and $W_R = \{x : P_{\vec{r}}(x) \geq c\}$;
    Update working set by $\mathcal{W} \leftarrow \mathcal{W} \cup \{W_L, W_R\}$;
  end while
  Add tree t_i to the ensemble $\mathcal{F} \leftarrow \mathcal{F} \cup \{t_i\}$;
end for
return($\mathcal{F}$);

3.2 Similarity kernel and clustering

We can now describe algorithms for the generation of a similarity kernel with *rpForests* and for clustering with the resulting similarity kernel.

Once *rpForests* is grown, the generation of the similarity kernel is fairly straightforward. For each tree, one scans through all the leaf nodes and collect information regarding if two points are in the same leaf node, and then aggregate such information from all trees in *rpForests*. For ease of description, let $S[A_1, A_2]$ denote all entries in matrix S with their position indexed by the Cartesian product $A_1 \times A_2$ of two sets of integers A_1 and A_2. The generation of the similarity kernel is described as Algorithm 2. Note that here the notation, $\mathcal{N}$, for a node may also refer to the index of all points in this node for ease of description.

Algorithm 2 *rpfSimilarity($\mathcal{F}$)*

```
Initialize a similarity matrix S ← 0;
for each tree t ∈ F do
    for each leaf node N ∈ t do
        Increase the similarity count for each entry in S[N,N];
    end for
end for
Set S ← S/(number of trees in F);
Return S;
```

The similarity matrix S as produced by Algorithm 2 is a valid kernel matrix. This can be argued as follows. Let $S^{(t)}$ denote the similarity matrix generated by tree t. Then $S^{(t)}$ is a block diagonal matrix with all points in the same leaf node form a diagonal block of all entries 1, and all off-diagonal blocks are 0 as those correspond to points from different leaf nodes. Let matrix M be one of the diagonal blocks in $S^{(t)}$. Then M is positive semidefinite, as the following holds

$$z^T M z = (z_1 + z_2 + \cdots + z_m)^2 \geq 0$$

for any vector $z = (z_1, ..., z_m)$. This implies that matrix $S^{(t)}$ is positive semidefinite. It follows that the average matrix

$$S = \frac{1}{T} \sum_{t=1}^{T} S^{(t)}$$

is also positive semidefinite. Thus the similarity matrix S produced by *rpForests* is a valid kernel matrix. Due to its intimate connection to *rpForests*, the resulting kernel is termed as *rpf-kernel*.

On rpf-kernel S, it is straightforward that one can apply spectral clustering to obtain a clustering of the original data. This is described as Algorithm 3. Note that here we threshold the kernel S following the same idea as [58]. This will

Algorithm 3 *rpfCluster(S, K)*

1: Threshold similarity kernel by $S_{ij} \leftarrow 0$ if $S_{ij} < \beta_1$;
2: $S \leftarrow exp(S/\beta_2)$ for some bandwidth β_2;
3: Apply spectral clustering to S to get the cluster assignment;

help get rid of the spurious similarity between points in different clusters (in the ideal case, points from different clusters would have a 0 similarity if clustering is concerned). Also similar as in the practice of using the Gaussian kernel in various kernel methods, we apply a bandwidth to reflect the correct scale at which the data are clustered.

There are several variants of spectral clustering around. In the present paper we adopt *normalized cuts* (Ncut) [50]. Ncut computes the second eigenvector of the Laplacian of matrix S to find a bipartition of the data. The nonnegative components of the eigenvector corresponds to one partition and the rest to the other. The same procedure is applied recursively until reaching the number of specified clusters. For more information about spectral clustering, readers can refer to [50,43,53].

4 Theoretical analysis

The growth of *rpForests* involves quite a bit of randomness, i.e., in the choice of splitting directions and the splitting point. A central concern would be the quality of the rpf-kernel learned by *rpForests*—will the learned similarity kernel preserve the true similarity values? Our analysis would give an affirmative answer to this. Due to space limit, we briefly summarize our results here.

Following a similar line of argument as in [62], we characterize the behavior of data points according to whether they are nearby or far-away. Here a crude rule for *nearby* is that one point is a k-nearest neighbor of another (or approximately, the two points are in the same tree leaf node) while *far-away* is when the distance between two points is larger than a small constant, say α; here both k and α are application dependent. Then, by a normal approximation on the number of times two points fall into the same tree leaf node in the *rpForests*, we show that, with high probability, any given pair of nearby points will have high similarity while the similarity of far-away points will stay below δ for small constant δ. This is a highly desirable property of the similarity matrix. In Section 5.1, we will provide a toy example to empirically demonstrate such a property.

5 Experiments

Our experiments consist of two parts. In the first part, we will give illustrative examples to help the readers better appreciate the desired property of the rpf-kernel learned by *rpForests*, and its relevance for clustering. In the second part, we evaluate the empirical clustering performance of *rpfCluster* and compare it to three competing clustering algorithms, including one of the most widely

used clustering algorithms, K-means clustering [39], as the baseline, the NJW algorithm [43] as a popular implementation of spectral clustering (commonly acknowledged as the class of best clustering algorithms), and CF [58] as state-of-the-art ensemble clustering algorithm, on a wide variety of real datasets. The two parts are presented in Section 5.1 and Section 5.2, respectively.

5.1 Illustrative examples

In this section, we will provide two illustrative examples. One serves to illustrate the desired property of similarity kernel learned by *rpForests*, and the other to demonstrate the propagation of similarity through points in the same cluster.

To appreciate the desirable property of the rpf-kernel produced by *rpForests*, we will use a popular yet simple dataset, the *Iris flower* data. It was introduced by one of the founders of modern statistics, R. A. Fisher, in 1936 for discriminant analysis, and has since become one of the most widely used datasets. The data consist of three species of Iris, *Iris setosa*, *Iris versicolor*, and *Iris virginica*, with 50 instances each on four features, the length and width of the sepals and petals, respectively.

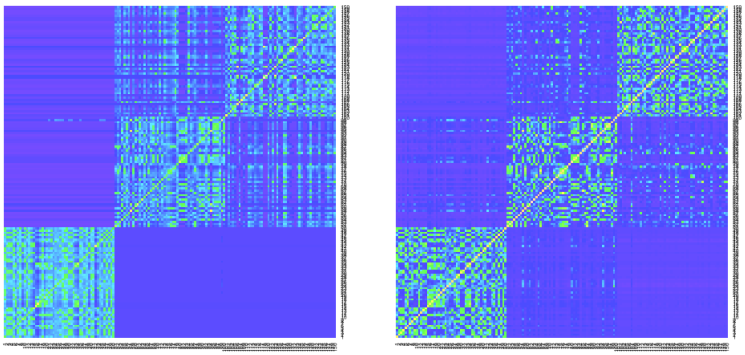


Fig. 1. *Heatmap of the similarity matrix generated by the Gaussian kernel (left) and by rpForests (right), respectively.*

Figure 1 shows the heatmap of the similarity matrix generated by the Gaussian kernel and by *rpForests*, respectively. It can be seen that, in both cases, the similarity between data points in the same species (i.e., the diagonal blocks) are higher than otherwise. In particular, the similarities are close to 0 between the *Iris setosa* and the *Iris virginica*. This is expected. However, the contrast between other diagonal and non-diagonal blocks by *rpForests* is much sharper than those by the Gaussian kernel. We attribute this to the Gaussian kernel as a sole function of the distance between points which is Euclidean and every feature is equally weighted, and further the potential smoothing effect of the Gaussian

kernel. In contrast, the kernel formed by *rpForests* would be a result of both the distance between points and their neighborhood thus is able to take advantage of the *geometry* in the data. Further work is being conducted to understand these.

We then run spectral clustering algorithm on the rpf-kernel generated by *rpForests*, and obtain a clustering and co-cluster accuracy (see definition in Section 5.2) of 96.67% and 94.95%, respectively. This is at the level of *classification* by some best classifiers such as RF despite that *rpfCluster* is unsupervised learning. The clustering and co-cluster accuracy on the Gaussian kernel are noticeably inferior, which are 92.00% and 90.55%, respectively.

When adopting the rpf-kernel produced by *rpForests* for clustering, one might notice that two points in the same cluster may have a low similarity (even though rpf-kernel is able to pick up structural information from the data) if they are far away from each other (since these two points are far away, likely they would be put into different leaf nodes by *rpForests* thus a low value of similarity in the resulting similarity kernel); for example, two points that are located at the opposite ends of the cluster. However, this will *not* cause a problem for the subsequent spectral clustering which is built on local similarity. One empirical evidence is various algorithms for the speeding up of spectral clustering by sparsifying the similarity matrix based on k-nearest neighbors where similarity between non-kNNs are truncated to be 0.

Here we supply a simple numerical example for illustration: far-away points will not be assigned to different clusters by spectral clustering as long as they are inside a region where all points has high similarity to their near neighbors. Assume there are 9 points, X_{1-9}, on a line which form two clusters, X_{1-4} and X_{5-9}, respectively. Assume, for each point, only its immediate neighbors have a non-zero similarity; further assume the similarity between X_4 and X_5 (from different clusters) is 0.3. The similarity matrix is given by

$$A = \left[\begin{array}{ccccccccc} 1.0 & 0.9 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.9 & 1.0 & 0.9 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.9 & 1.0 & 0.9 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.9 & 1.0 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 \\ \hdashline 0.0 & 0.0 & 0.0 & 0.3 & 1.0 & 0.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 & 0.9 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 & 0.9 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 & 0.9 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 \end{array}\right]$$

The eigenvector used for spectral clustering (normalized cuts [50]) is given by

$$[0.340\ 0.469\ 0.392\ 0.219\ -0.098\ -0.249\ -0.351\ -0.419\ -0.304]^T,$$

which gives the expected clustering, i.e., positive components correspond to one cluster and the rest the other cluster.

5.2 Experiments on real datasets

A wide range of real data are used for performance assessment. This includes 11 benchmark datasets taken from the UC Irvine Machine Learning Repository [34],

namely, Soybean, SPECT Heart, image segmentation (ImgSeg), Heart, Wine, Wisconsin breast cancer (WDBC), robot execution failure (lp5), Madelon, Musk, Naval Plants, and the Magic Gamma (mGamma) dataset, as well as a remote sensing dataset (RS) [60], totally 12 datasets. For WDBC, it is standardized on its {5, 6, 25, 26}-th features; the Wine dataset is standardized on its {6, 14}-th features; for the Naval Plants data, we treat any record of measurements as requiring maintenance if both the q3Compressor and q3Turbine variables are above their median values thus converting the original numerical values into categorical labels. A summary of the datasets is given in Table 1. Note that all datasets come with labels. We made such a choice by recognizing that the ultimate goal of clustering is to get the membership of all the points right; many existing metrics for evaluating clustering algorithms are often a surrogate of this due to the lack of true labels.

Table 1. *A summary of datasets.*

Dataset	Features	Classes	#Instances
Soybean	35	4	47
SPECT	22	2	267
ImgSeg	19	7	2100
Heart	13	2	270
Wine	13	3	178
WDBC	30	2	569
Robot	90	5	164
Madelon	500	2	2000
RS	56	7	3303
Musk	166	2	6598
NavalPlants	16	2	11934
mGamma	10	2	19020

Two different performance metrics are used; these are adopted from [58]. One is the clustering accuracy, and the other is the co-cluster accuracy. *Clustering accuracy* is the percent of data points that receive the correct cluster membership assignment according to the true label, subject to permutations on the labels. *Co-clustering accuracy* is the percent of correctly clustered pairs out of all possible pairs, where by *correctly clustered pair* we mean two data points, determined to be in the same cluster by a clustering algorithm, are also in the same cluster according to their true labels. Having different performance metrics allows to assess a clustering algorithm from different perspectives since one metric may favor certain aspects while overlooking others. Using the clustering or co-cluster accuracy has the advantage of closely aligning to the ultimate goal of clustering—assigning data points to proper groups—while other metrics are often a surrogate of the cluster membership.

We compare *rpfCluster* to K-means clustering [39], the NJW algorithm [43], and CF [58]. The NJW algorithm is a popular variant of spectral clustering. It works on the eigen-decomposition of the Laplacian of the similarity matrix over the data, followed by a k-means clustering on the embedding of the original data points by the top few eigenvectors. CF is a clustering ensemble algorithm where each clustering instance is generated by randomized feature pursuit according to the κ criterion [58]. For more details about the algorithm and its parameter settings, please refer to [58] from which we also adopt some experimental results in our comparison. The comparison to NJW and CF can also be viewed on different similarity kernels. The similarity kernel in CF is learned by randomized feature pursuits in individual clustering instances and then aggregate. In NJW, the Gaussian kernel is used by default.

For K-means clustering, the R package *kmeans()* was used with the "Hartigan-Wong" initialization, and the two parameters (n_{it}, n_{rst}), which stands for the maximum number of iterations and the number of restarts during each run, respectively, are set to be $(1000, 100)$. For NJW, function *specc()* of the R package "kernlab" [30] was used with the Gaussian kernel and an automatic search of the local bandwidth parameters. The number of trees in *rpForests* are chosen from $\{200, 400, 600\}$ and the difference in results is very small, the node splitting constant n_s is 30 except for 12 for Soybean and 200 for Madelon, the threshold level β_1 is chosen from $\{0, 0.1, 0.2, 0.3, 0.4\}$, and the step size for the search of bandwidth β_2 is 0.01 within (0,1] while 0.1 over (1,200].

The results on clustering accuracy and co-clustering accuracy are shown as Figure 2 and Figure 3, respectively. On all but two of the 12 datasets, either

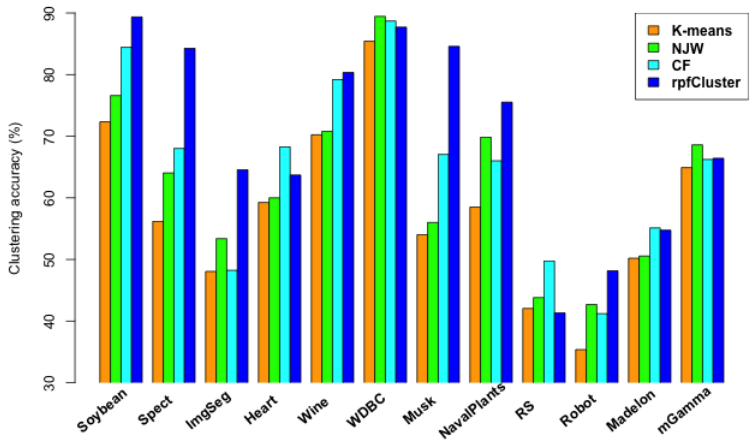


Fig. 2. *Comparison between K-means clustering, spectral clustering (NJW), CF, and rpfCluster for clustering accuracy.*

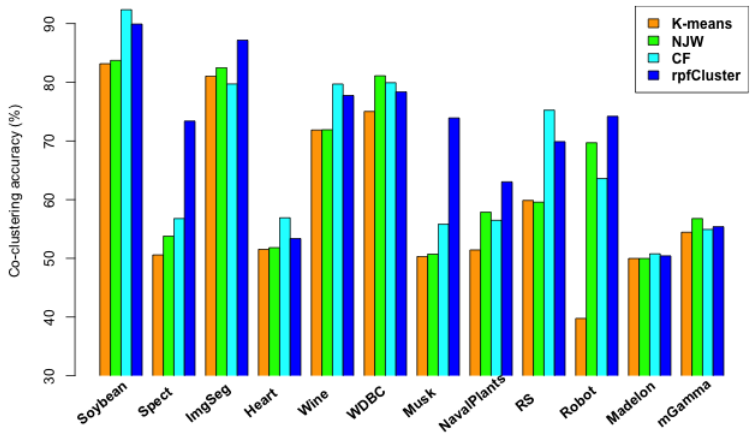


Fig. 3. *Comparison between K-means clustering, spectral clustering (NJW), CF, and rpfCluster for co-clustering accuracy.*

rpfCluster or CF is leading with *rpfCluster* having an edge. Under clustering accuracy, *rpfCluster* is leading on 7 datasets, and ranks the second on 3 datasets while CF leads on 3 and seconds on 5. For co-cluster accuracy, *rpfCluster* leads on 5 datasets and seconds on 6 while CF leads on 5 and seconds on 3. Overall, *rpfCluster* outperforms CF (also NJW and K-means clustering) on both of the two performance metrics.

6 Conclusions

We have proposed an effective approach for the unsupervised learning of a similarity kernel by *rpForests*. Our approach combines the power of ensemble methodology and the flexibility of trees. It is simple to implement, and readily adapt to the geometry of the underlying data. Our theoretical analysis reveals highly desirable property of the learned rpf-kernel: *far-away points have low similarity while high similarity for nearby points*, and the similarities have a native interpretation as the probability of points staying in the same tree leaf nodes during the growth of *rpForests*. The learned rpf-kernel is readily incorporated into our clustering algorithm *rpfCluster*. On a wide variety of real and benchmark datasets, *rpfCluster* compares favorably to spectral clustering and a state-of-the-art clustering ensemble algorithm. Given the desirable theoretical property and the highly competitive empirical performance on clustering, we expect rpf-kernel to be applicable to other problems of an unsupervised nature or as a regularizer in supervised or weakly supervised settings.

References

1. T. W. Anderson. *An Introduction to Multivariate Statistical Analysis.* John Wiley & Sons, 1958.
2. F. Angiulli and C. Pizzu. Fast outlier detection in high dimensional spaces. *Lecture Notes in Computer Science*, 2431:43–78, 2002.
3. F. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2003.
4. A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.
5. A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv:1306.6709.v4*, 2014.
6. J. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
7. P. J. Bickel and L. Breiman. Sums of functions of nearest neighbor distances, moment bounds, limit theorems and a goodness of fit test. *The Annals of Probability*, 11(1):185–214, 1983.
8. P. J. Bickel and D. Yan. Sparsity and the possibility of inference. *Sankhya: The Indian Journal of Statistics, Series A (2008-)*, 70(1):1–24, 2008.
9. L. Bobrowski. Ranked Modeling with feature selection based on the CPL criterion functions. In *Proceedings of the 4th international conference on Machine Learning and Data Mining in Pattern Recognition*, pages 218–227, 2005.
10. L. Bobrowski. Class separating measures of similarity for the CBR scheme. In *Proceedings of the Industrial Conference on Data Mining*, pages 5–18, 2012.
11. T. Boongoen and N. Iam-On. Cluster ensembles: A survey of approaches with recent extensions and applications. *Computer Science Review*, 28:1–25, 2018.
12. L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.
13. V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *Technical Report, University of Minnesota*, 2007.
14. O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 15*, pages 601–608, 2003.
15. G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.
16. C. Cortes and V. N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
17. S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Fortieth ACM Symposium on Theory of Computing (STOC)*, 2008.
18. S. Dasgupta and K. Sinha. Randomized partition trees for nearest neighbor search. *Journal Algorithmica*, 72(1):237–263, 2015.
19. J. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216, 2007.
20. I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM international conference on Knowledge discovery and data mining (SIGKDD)*, 2004.
21. Z. Ding, M. Shao, and Y. Fu. Robust multi-view representation: A unified perspective from multi-view learning to domain adaption. In *Proceedings of 27th International Joint Conference on Artificial Intelligence*, pages 5434–5440, 2018.

22. W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, 2011.
23. X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.
24. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, 1996.
25. J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding the best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.
26. A. Gionis, H. Mannila, and P. Tsaparas. Cluster aggregation. In *the 21st International Conference on Data Engineering (ICDE)*, 2005.
27. M. Hirzer. Large scale metric learning from equivalence constraints. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2288–2295, 2012.
28. T. Hofmann, B. Schölkopf, and A. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.
29. Z. Kang, C. Peng, and Q. Cheng. Kernel-driven similarity learning. *Neurocomputing*, 267(C):210–219, 2017.
30. A. Karatzoglou, A. Smola, and K. Hornik. kernlab: Kernel-based Machine Learning Lab. http://cran.r-project.org/web/packages/kernlab/index.html, 2013.
31. M. Kleinbort, O. Salzman, and D. Halperin. Efficient high-quality motion planning by fast all-pairs r-nearest-neighbors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2985–2990, 2015.
32. B. Kulis. Metric Learning: A Survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2015.
33. E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17*, 2005.
34. M. Lichman. UC Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml, 2013.
35. D. Lim and G. Lanckriet. Efficient learning of mahalanobis metrics for ranking. In *Proceedings of the 13rd International Conference on Machine Learning (ICML)*, 2014.
36. K. Liu, A. Bellet, and F. Sha. Similarity learning for high-dimensional sparse data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.
37. T. Liu, A. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Neural Information Processing Systems (NIPS)*, volume 19, pages 825–832, 2004.
38. W. Liu, C. Mu, R. Ji, S. Ma, J. R. Smith, and S.-F. Chang. Low-rank similarity metric learning in high dimensions. In *Proceedings of the 29-th AAAI Conference on Artificial Intelligence*, pages 2792–2799, 2015.
39. S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(1):128–137, 1982.
40. Y. P. Mack and M. Rosenblatt. Multivariate k-nearest neighbor density estimates. *Journal of Multivariate Analysis*, 9:1–15, 1979.
41. M. Meila, S. Shortreed, and L. Xu. Regularized spectral learning. Technical report, Department of Statistics, University of Washington, 2005.

42. P. Moutafis, M. Leng, and I. A. Kakadiaris. An overview and empirical comparison of distance metric learning methods. *IEEE Transactions on Cybernetics*, 47(3):612–625, 2017.
43. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Neural Information Processing Systems (NIPS)*, volume 14, 2002.
44. C. Otto, D. Wang, and A. K. Jain. Clustering millions of faces by identity. *arXiv:1604.00989*, 2016.
45. S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD*, pages 427–438, 2000.
46. B. Schölkopf. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
47. B. Schölkopf and A. Smola. *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.
48. M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.
49. C. Shahabi and D. Yan. Real-time pattern isolation and recognition over immersive sensor data streams. In *Proceedings of the 9th Int'l Conference on Multi-Media Modeling*, pages 93–113, 2003.
50. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
51. T. Shi and S. Horvath. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138, 2006.
52. A. Strehl and J. Ghosh. Cluster ensembles – a knowlwdge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:582–617, 2002.
53. U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, pages 395–416, 2007.
54. B. Wang, A. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and A. Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods*, 11:333–337, 2014.
55. K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.
56. E. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 521–528, 2002.
57. C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *arXiv:1304.5634*, 2013.
58. D. Yan, A. Chen, and M. I. Jordan. Cluster Forests. *Computational Statistics and Data Analysis*, 66:178–192, 2013.
59. D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD*, pages 907–916, 2009.
60. D. Yan, C. Li, N. Cong, L. Yu, and P. Gong. A structured approach to the analysis of remote sensing images. *International Journal of Remote Sensing*, 40(20):7874–7897, 2019.
61. D. Yan, T. W. Randolph, J. Zou, and P. Gong. Incorporating deep features in the analysis of tissue microarray images. *Statistics and Its Interface*, 12(2):283–293, 2019.
62. D. Yan, Y. Wang, J. Wang, H. Wang, and Z. Li. K-nearest neighbor search by random projection forests. *IEEE Transactions on Big Data*, PP:1–12, 2019.
63. L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2, 2006.

Automated Topic modelling for Unlabelled Network Data*

Sadaf Azad[1][0000−1111−2222−3333], Ci Lei[2][1111−2222−3333−4444], David Farrar[3][2222−3333−4444−5555], and Thomas Mangin[4][3333−4444−5555−6666]

[1] University of Bradford, West Yorkshire, UK
s.azad4@bradford.ac.uk
[2] University of Bradford, West Yorkshire, UK
c.lei1@bradford.ac.uk
[3] Exa Networks, Bradford, West Yorkshire, UK
david.farrar@exa.net.uk
[4] Exa Networks, Bradford, West Yorkshire, UK
thomas.mangin@exa.net.uk

Abstract. Training text classifiers for unlabelled data is a big challenge. Topic models are used to produce a summary of the topics describing a collection of documents. However, the decision about the correct number of topics and their assignment is ultimately based on human judgement. This requires detailed domain knowledge and expertise. The aim of this research is to utilize topic models as part of an unsupervised approach to produce labelled data without human intervention. The novelty of this study is that we have devised a completely automated approach to select the best describing model for our use case(web network data). The principles of graph theory and topic modelling are utilized to generate a labelled data set for the use case. Discriminative models are used on top of the unsupervised topic models to ensure the selection of the model with an optimal number of topics. The existing evaluation measures for selecting the best topic model have also been explored while understanding and devising a novel quantitative approach to select the optimal model.

Keywords: Unlabelled Data· Networks · Topic modelling· Text classification · Evaluation Metrics· Graph Theory · Naive Bayes

1 Introduction

Web is a huge domain and currently there are 1.5 billion registered websites. In dealing with content on the web, the biggest challenge is to handle the unstructured and unlabelled text data. Labelling data requires a lot of human effort but still it is errorsome. In most cases of web data categorization, labelling is performed by humans prior to training the models for text classification. DMOZ and Yahoo are the best examples of human labelled data[9, 14]. Using human

* Supported by Exa Networks, Innovate UK.

effort to match with the ever-evolving growth of web is not feasible in terms of effort, time and money. The major contribution of our research is that network attributes as well as textual attributes of web were considered to generate a labelled data set. A totally unsupervised approach is adapted to find the clusters of similar webpages based on their connectivity while automating the process of labelling through the use of multiple topic models. Instead of domain expert, machine learning based discriminative models were used to select the best topic model for labelling the data.A Three-Level strategy was defined to generate labelled network data.

First level of research is to investigate the network topology of the web and to generate a dataset. In most networks connectivity between nodes implies correlation in terms of their attributes. However, in case of web graph connectivity does not always imply content similarity. We can expect to find Categorical-Variety in a densely linked community of webpages. Therefore, labels for the nodes in the graph cannot be generated by solely relying on connectivity. By utilizing the structural aspects (topology) of web, a dataset of densely linked domain level webpages was generated. This ensured that we have big enough unlabelled dataset with groups of webpages having topical similarity. Louvain, a node clustering algorithm that clusters the nodes in a graph by optimizing the modularity was utilized to generate the data[5].

The second phase of the research was focused on understanding the HTML structure and capturing the most descriptive feature of the web pages. The aim of this phase is to add descriptive features to the nodes in the data. Web pages are formatted using Hypertext-Mark-up-Language (HTML) and Meta-Tags are the best explanatory features of the web pages. The Meta-tags including title and description are the first level of information that are visible through search engines results[20]. For experimental purpose, a Web-Crawler was developed to extract the title tags of the web pages. Features were extracted from the title tags by defining a preprocessing pipeline. Textual features from the titles were extracted as single and collocated tokens to construct the term vector representations of the web pages in the data set[15, 23].

Third phase of the research involved establishing a model to uncover the topics of the descriptive titles of the web pages. The focus of the phase is to generate labels for the network nodes in our data set based on the novel automated approach. The suggested model is based on a combination of generative and discriminative models. Latent dirichlet allocation, which is an unsupervised generative statistical topic model was used to generate topics. The model aims to optimize the probability of the latent topics by incorporating the information from the Document-Term and Topic-Term matrices[4]. The biggest challenge in any generative model like LDA is to define the k which is the optimal number of clusters or topics. Various metrics and their extensions have been suggested to estimate the number of topics in a collection of documents including coherence score, perplexity, held-out likelihood. However different metrics suggest different number of optimal topics. Even same metric can produce different results in different runs. In most cases, the uncertainty associated with selection of number

of topics is handled by human intervention. Topic models that produce topics matching with human judgments are selected as the best models. Unlabelled documents are fed to the model in the form of vectors that are based on the frequency of the individual tokens and collocations[15, 23].

This challenge is addressed by instigating a quantitative approach to model the qualitative responses produced by various topic models. The discriminative models (classifiers) are trained by feeding the labelled data produced by topic model. The discriminative models are then tested on the test data. Evaluation metrics like accuracy, precision and recall are then measured to check the performance of the discriminative models. Topics from the topic model having the best evaluation scores through discriminative modelling are then selected as the assigned topics to the data. Our approach results in automating the whole process while selecting the optimal topic model. Various topic models were generated with different number of topics (k). Classifiers were trained on the data produced by different topic models. The classifier that produced best accuracy was selected. This selection of the classifier also solved the estimation problem that is normally associated with selecting the right number of topics in topic models.

In Section 2 of the paper, we have described the background and related work for community detection in networks. We have also discussed about the topic modelling algorithms particularly Latent Dirichlet Allocation in detail. We have discussed the limitations of LDA as an unsupervised model in detecting the right number of topics. In Section 3, we have proposed a methodology to address the problem of finding the right number of topics. Our methodology suggests using a combination of generative and deterministic models to select the topic model with higher performance over data classification. In Section 4, we have described in detail the use case and the proposed model. Section 5 is about the experimental work and the results. In Section 6, we have discussed future work and conclusion.

2 Background and Related Work

Labelling network data is a key characteristic of networks with practical and analytical applications. Glass, Rendezvous algorithm and DeepWalk are some of the Semi-Supervised techniques for labelling the nodes of the network [10, 3, 18]. The Glass method and Rendezvous algorithms calculate the absorption probabilities of the labels from labelled nodes to unlabelled nodes. These probabilities are calculated by generating random walks in the graph modelled as discrete time Markov chain. The probabilities are then used to derive a distribution of labels over the missing labels of the associated nodes in the network [10, 3]. DeepWalk models unsupervised learning from sequence of words to graphs. The random walks are generated and the originated patterns in the walks are treated as sentences which are then used to learn latent representations of nodes in the graph. Clusters of nodes are generated based on similarity of their representations. These representations are a mean to perform multi-label classification [18]. All these approaches rely on some prior information about the labelling of the nodes. Also, these methods are used in networks where connectivity implies

similarity in all aspects. However, in cases where similarity does not rely on connectivity only; other measures need to be identified. For example, in web graph, there is a higher probability that two websites that are very similar to each other in terms of content may not be part of the same community as they never happened to be topologically linked with each other. The focus of the current research is to address the situation where the graph is totally unlabelled.

Communities in the graphs can be detected using graph theory algorithms like Louvain or deep learning algorithms like node2vec [5, 12]. Louvain extracts communities of connected nodes based on modularity optimization where the modularity is optimized as:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in} -}{2m} \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \quad (1)$$

where Σ_{in} is the sum of weights of the edges in the community, Σ_{tot} is the sum of the weights incident to node i , $k_{i,in}$ is the sum of weights of the links from i to nodes in the community and m is the sum of weights of all the links in the community. The process of community detection is recursive where nodes are removed and added between the neighboring communities. The nodes are assigned to the community for which the maximum modularity is achieved.
The data set in our research was generated using Louvain modularity optimization on a bigger web network[5, 8]. As the nodes in the data were unlabelled and only link information was present for the network; therefore, the URLs of the web pages were crawled and titles were extracted from the Meta-Tags. Titles are the best descriptive features that are added by the website owners to the HTML headers of their websites to describe their domain in a short context. The main part of the research is focused on topic modelling and defining an evaluation measure for topic models.
Topic models are the set of unsupervised algorithms that are used to group relative texts while highlighting the summary of the concepts contained within those documents. Different approaches for topic modelling exist [13, 4]. The input to the topic models is the documents that are presented as matrices containing terms distribution over the documents. The input matrices are constructed by representing textual features as frequencies or weighted frequencies. Moreover, the features could be extracted from documents by applying different filters depending upon the domain knowledge related to those documents. Keywords, named entities, bigrams etc. are some of the features that can be used to construct input matrices for topic models.

LDA (Latent Dirichlet Allocation) is the probabilistic, generative text data topic model that is based on Three-Level hierarchical Bayesian probability models. The model is a standardized version of probabilistic latent semantic indexing and it aims to generate compressive representations of the documents by mapping the distribution of words with respect to the k latent topic variables [13]. As shown in figure (1), the three layers of the model are document collection layer, individual document layer and word layer. The document collection layer is defined by α and β which are the relative strength among latent topics and

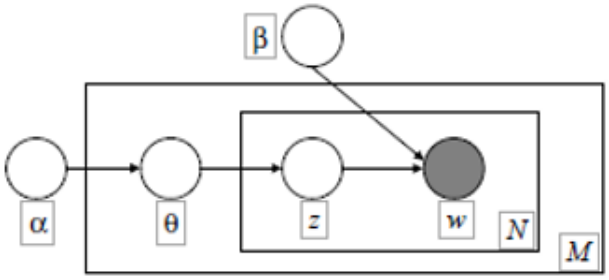


Fig. 1. LDA Model Diagram as Directed Graph with 3-Layers.

probability distribution of latent topics respectively. The document layer is defined by θ which is the joint distribution of a topic mixture, and the word level layer is defined by z and w which are the set of N topics and set of N words respectively. The generative LDA model solves the posterior of the hidden topic variables for the target documents through distribution of words over documents and topics.

LDA was preferred to model the topics for our data because this model has complete document generation process and no information is truncated in the topic generation procedure. Due to being hierarchical model, LDA is more stable and is not prone to overfit. LDA with Gibbs sampling uses the parameters of the multinomial distribution θ and β that are integrated out while keeping the latent counts z for the topic distribution[19].

2.1 How many topics?

The major issue with LDA is the correct inference for the number of latent topic variables in a collection of documents. Coherence and perplexity are the intrinsic measures that are used to obtain the optimal number of topics for the models. Coherence score is the measure of the degree of semantic similarity between high scoring words in the documents. The coherence score among set of topic words, W is calculated as the pairwise distributional similarity among them as:

$$coherence\,(W) = \sum_{(w_i,w_j)\epsilon W} score\,(w_i, w_j, \epsilon) \tag{2}$$

where W is the set of words constituting a topic and ϵ is the smoothing function ensuring the return of real number. The score is calculated as $log\frac{D(w_i,w_j)+\epsilon}{D(w_i)}$. $D\,(w_i, w_j)$ is the count of number of documents containing words i and j and $D\,(w_i)$ is the count of the documents containing word i. Higher coherence score indicate that words constituting a topic are semantically related and hence the topic is of good quality [21].

Perplexity is the statistical evaluation of topic quality and measures how well a probability model predicts a sample. It represents the confusion in assignment

of a topic to a document [24]. For a document collection $D_{test} = \{w_d\}$, it is defined as reciprocal geometric mean of the likelihood of the given document set. The model is represented as:

$$Perplexity(D_{test}) = exp\left\{-\frac{\sum_d logp\,(w_d)}{\sum_d N_d}\right\} \tag{3}$$

Using measures like coherence and perplexity can be used to learn the optimal number of topics by adapting a heuristic approach. A set of topic models is produced ranging from 1 to k as t=1,..,k and the optimal topic model k* is:

$$k* = argmax_k \frac{1}{k} \sum_{t=1}^{k} coherence_{topic} \tag{4}$$

$$k* = argmin_{\ k} \frac{1}{k} \sum_{t=1}^{k} perplexity_{topic} \tag{5}$$

Although these approaches seem to suggest optimal topic, but it has been found that the results of these measures are not reliable and do not correlated with semantically interpretable topics. It requires human effort to select the optimal number of topics which is time consuming and requires in depth domain knowledge. Other approaches to produce more coherent topics involve aggregation of LDA and LSA (latent semantic analysis). The conceptual vectors generated by LSA can be used to perform topic modelling through LDA. This results in more coherent topics in a model by relying on the abstract features of documents[22]. However, the main focus of research related to LDA has been towards increasing the coherence among topics or using human labelled data to guide the model to guess the right number of topics[16]. The selection of optimal number of topics is still based on qualitative judgment through manual inspection by domain experts [6].

This gap in research related to finding correct number of topics in a collection of unlabelled documents has led us to define our methodology which is based on incorporating supervised machine learning models on top of the results produced by topic models for different values of k.

3 Methodology

Our methodology for finding the optimal number of topics is based on two stages and it incorporates the use of discriminative models. The discriminative models are used to solve the problem of optimal number of topics while quantitatively evaluating the performance of the qualitative results produced by a range of topic models for a documents corpus.

The set of documents is:

$$D = d_1, d_2, d_3, ..., d_N \tag{6}$$

where $d_1, d_2, ...$ are the individual documents in the corpus.

$$V = w_1, w_2, ..., w_v \tag{7}$$

is the vocabulary constituting the words in the corpus. The frequency of the term $w \in V$ for the document $d_i \in D$ is $f_{d_i}(w)$. The term vector for document d_i is denoted by:

$$\vec{t}_{d_i} = (f_{d_i}(w_1),\ f_{d_i}(w_2), ..., f_{d_i}(w_v)) \tag{8}$$

Let T be the collection of many LDA topic models produced on the set D with topics ranging from 2 to n. For any single topic model $T_K = \theta_1, \ldots, \theta_K$, the documents in D are assigned the dominant θ_i. For any given document d_i, the dominant θ_i with highest proportion of words present in the document is selected. The class label is assigned to a single document d_i as:

$$c_{d_i} = \theta_i \tag{9}$$

where θ_i contains the highest proportion of words for a given document d_i based on its distribution of words. The outcome from the assignment of dominant topics in $T_K \epsilon\ T$ result in a set of real valued labels representing as:

$$C_{T_k} = \{c_1,\ c_2,\ \ldots, c_k\ \} \tag{10}$$

where c_i is the class assignment for d_i in the document set $D = d_1,\ d_2, ..., d_D$.

At the second stage, the discriminative power of the classifiers was used recursively to find the topic model in T giving the best accuracy. The aim of the classifier is to map the test document represented by $\vec{t}_{d_{test}}$ to C_{T_k} based on the training set. For all the topic models, accuracy, precision and recall were evaluated on the test set and the $T_K \in T$ having maximum accuracy and precision was selected as the best topic model.

4 Use Case and The Model

The use case for automatically finding the optimal topic model through our strategy comprise of collection of highly connected webpages. The data set was produced through application of community detection algorithms on the web graph. The webpages are unlabelled in the data and the only attribute for generating that data was the link information between the webpages. The motive behind selection of web data was that despite being connected, there is a tendency for the presence of dissimilarity between them based on their textual attributes. The dissimilarity forms the basis to be able to find different yet concentrated topics within the data collection. To the best of our knowledge there is no such model where webpages are automatically labelled using a combination of topic models and classifiers without prior information of their labels. Our work is the first of its kind.

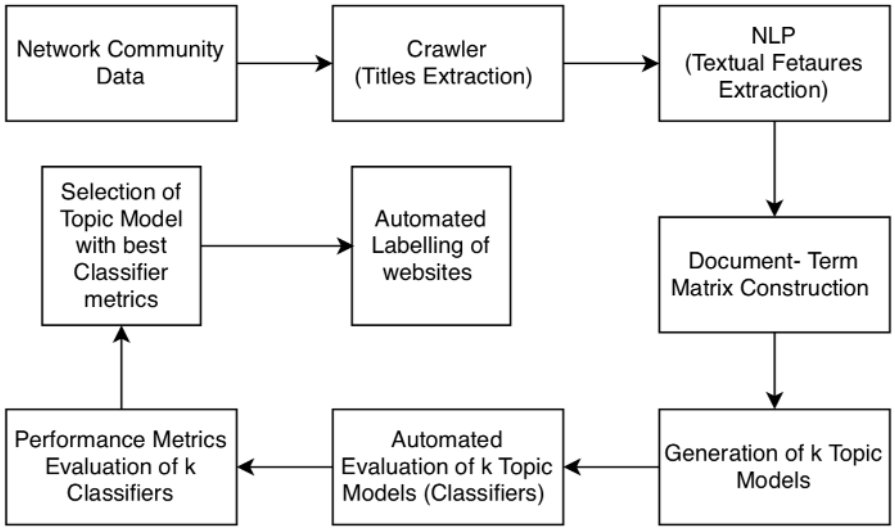


Fig. 2. Process Flow Diagram for the Automated Classifier

The process flow diagram shows that starting from the network data consisting of only edges information between web pages in the form of $A- > B$ i.e. a directed graph, a crawler was defined to extract the titles of the web pages as the minimal textual attributes. The titles were extracted through utilizing a customized crawler from the HTML structure ¡head¿ of the webpages by exploiting the domain knowledge of the web.
Textual features were then extracted from the title tags of the webpages by defining a preprocessing pipeline. The pipeline comprised of using regular expressions

to exclude special characters; language detection model to detect English language; collocations to extract single and collocated pair of words. Stop-words were removed and the extracted features were then lower-cased and lemmatized to ensure extraction of useful keywords while removing the noise. Stop-words and other special characters were considered as noise in the model.

The "bag-of-word" (Vector Space Model) approach was adapted to create vectorized document representations. In bag of words, the order or the structure of the document is not important to construct the feature representations for the documents[23].

The extracted tokens were used to prepare the corpus in the form of Document-term matrix to train and evaluate the topic models. Topic models ranging from 2 to 20 were produced. The LDA topic models were evaluated through supervised machine learning classifiers. Class imbalance was addressed by selecting the classifiers that balanced the classes to solve the problem of bias created due to dominant class in the training data [7, 17]. The model giving the highest accuracy and precision in the train-test scenario was selected and the topics were assigned to the documents as labels. The selected topics were also in agreement with human expert evaluator.

5 Experiments and Results

Text from title tags was crawled for 6394 websites forming a community network. The text was processed through the preprocessing pipeline as described earlier to create feature vectors for the websites. A dictionary was constructed for all the features in the text corpus based on the frequencies of the tokens. The tokens were mainly the keywords highlighting the context of the web pages. Fig 3 shows the word cloud generated from the corpus included in the research.

Fig. 3. Word Cloud generated from the Corpus of Web pages Titles

Based on the document-term matrix produced from the corpus, various topic models were trained ranging from 2 to 20. To strengthen the evaluative analysis of the topic models and to learn the optimal number of topics, intrinsic measures including semantic coherence, and held-out likelihood i.e. perplexity of various topic models were evaluated. Deciding the optimal topic was not possible based

on these measures as it can be seen from the plots of coherence and perplexity in Fig 4.

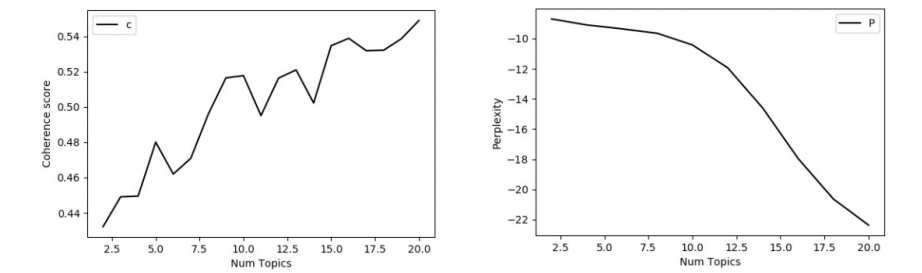


Fig. 4. Coherence and Perplexity measurements for various topic models ranging from 2-20.

To automate the selection of optimal model we adapted the methodology described in section 3 and trained Naïve Bayes on imbalanced data generated through topic models. The data was divided into train and test sets in a ratio of 80 and 20. The accuracy of Naïve Bayes classifier was highest for k=2 and was equal to 79.87%.

Table 1. Results generated by classifiers for different topic models. Topic model with k=2 has highest accuracy and F1-scores.

Topics	Accuracy			F1- Score		
	N. Bayes	Balanced RF	Balanced Bag	N. Bayes	Balanced RF	Balanced Bag
2	0.79870	0.68215	0.75713	0.79879	0.69935	0.75736
3	0.70171	0.53382	0.64384	0.70171	0.53382	0.64384
4	0.65852	0.53545	0.58191	0.65852	0.53482	0.58089
5	0.65362	0.47188	0.59331	0.65416	0.46977	0.59256
6	0.64222	0.44173	0.53871	0.64206	0.42531	0.53642
7	0.60309	0.44987	0.52078	0.60600	0.41746	0.51932
8	0.59658	0.43847	0.54523	0.60371	0.41382	0.54581
9	0.61369	0.37897	0.51589	0.61752	0.37630	0.51640
10	0.58517	0.38957	0.50041	0.59053	0.37281	0.49860
11	0.60717	0.38386	0.495517	0.60937	0.34992	0.48928
12	0.60717	0.43439	0.52812	0.61246	0.43035	0.52125
13	0.57212	0.420537	0.50285	0.57944	0.40044	0.49584
14	0.59658	0.41728	0.50041	0.60374	0.40354	0.50314
15	0.56886	0.39038	0.51996	0.5767	0.37836	0.51944
16	0.56398	0.41157	0.49144	0.57571	0.41445	0.49007
17	0.58598	0.41564	0.51670	0.59347	0.39588	0.51355
18	0.56153	0.41646	0.49959	0.57382	0.40405	0.49896
19	0.59983	0.41972	0.52078	0.60752	0.41134	0.51796
20	0.6031	0.37001	0.52567	0.61293	0.33521	0.52465

Balanced random forest and balanced bagging classifiers were used to balance the data and removing the bias due to the dominant class in the data. The balanced random forest artificially altered the class distribution by combining ensemble learning with down sampling of the majority class to create balance between all the classes in the classifier. The highest accuracy was 68.2% for k=2. Results were also recorded for balanced bagging classifier that balances the training set at fit time using a random under-sampler. Accuracy of 75.7% was recorded for topic models k=2 by using the balanced bagging classifier as shown in Table 1. Both F-1 score and accuracy score were highest for k=2 [11]. Although, there was some boost in accuracy at other values of k but the accuracy scores and the investigation of the word distribution in the topics suggested that topic model with two topics produced best describing topics for the use case.
The technical accuracy of the approach was evaluated by using a labelled benchmark dataset[2, 1]. The data set has two classes. However, our model was kept ignorant of the existing class labels. The methodology described in Section 3 was applied to the unlabelled data and it was found that out of various LDA models, the discriminative models produced best accuracy and f-1 scores for k = 2. For k =2, the Naive Bayes classifier produced highest accuracy of 81.5% while balanced random forest and balanced bagging models produced 76.7% and 81.5% accuracy respectively.

6 Discussion & Conclusion

Topic modelling is an appropriate approach to find the distribution of topics and words in a set of documents. However, in most cases, prior knowledge does not exist about the domain and the optimal number of topics. Existing evaluation metrics are a good measure to learn about the quality of topics produced by topic models. But, using these metrics to find the optimal number of topics is confusing and requires human inspection and evaluation to decide the right number of topics in a trial and error scenario. Our approach of combining the deterministic models with topic models produced promising results. The adapted methodology is a mean to reduce human involvement by automating the whole process of optimal topic selection and assignment to the documents. From use-case point of view, the methodology described to generate a data set of clustered documents based on network topology is also novel and can be utilized in situations where little is known about the data. The approach can be extended to produce labelled data sets for any kind of networks or unlabelled data sets where features can be represented in vectors and no prior information about the data labeling exists.

7 Acknowledgement

This work was funded by Innovate UK and Exa Networks as part of Knowledge Transfer Partnership program with University of Bradford.

References

1. Alberto, T.C., Lochter, J.V., Almeida, T.A.: Tubespam: Comment spam filtering on youtube. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). pp. 138–143. IEEE (2015)
2. Alberto, T.C., Lochter, J.V., Almeida, T.A.: UCI machine learning repository (2017), https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection
3. Azran, A.: The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In: Proceedings of the 24th international conference on Machine learning. pp. 49–56. ACM (2007)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
6. Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J.L., Blei, D.M.: Reading tea leaves: How humans interpret topic models. In: Advances in neural information processing systems. pp. 288–296 (2009)
7. Chen, C., Liaw, A., Breiman, L., et al.: Using random forest to learn imbalanced data. University of California, Berkeley **110**(1-12), 24 (2004)
8. Crawl, C.: Common crawl. URl: http://http://commoncrawl. org (2020)
9. DMOZ, D.: Open directory project (2002)

10. Glonek, M., Tuke, J., Mitchell, L., Bean, N.: Glass: Semi-supervised graph labelling with markov random walks to absorption. In: International Conference on Complex Networks and their Applications. pp. 304–315. Springer (2018)
11. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: European Conference on Information Retrieval. pp. 345–359. Springer (2005)
12. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
13. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 50–57 (1999)
14. Labrou, Y., Finin, T.: Yahoo! as an ontology: using yahoo! categories to describe documents. In: Proceedings of the eighth international conference on Information and knowledge management. pp. 180–187 (1999)
15. Nesselhauf, N.: Collocations in a learner corpus. John Benjamins Amsterdam (2005)
16. Newman, D., Bonilla, E.V., Buntine, W.: Improving topic coherence with regularized topic models. In: Advances in neural information processing systems. pp. 496–504 (2011)
17. Oza, N.C.: Online bagging and boosting. In: 2005 IEEE international conference on systems, man and cybernetics. vol. 3, pp. 2340–2345. Ieee (2005)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
19. Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., Welling, M.: Fast collapsed gibbs sampling for latent dirichlet allocation. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 569–577 (2008)
20. Raggett, D., Le Hors, A., Jacobs, I., et al.: Html 4.01 specification. W3C recommendation **24** (1999)
21. Röder, M., Both, A., Hinneburg, A.: Exploring the space of topic coherence measures. In: Proceedings of the eighth ACM international conference on Web search and data mining. pp. 399–408 (2015)
22. Stevens, K., Kegelmeyer, P., Andrzejewski, D., Buttler, D.: Exploring topic coherence over many models and many topics. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 952–961. Association for Computational Linguistics (2012)
23. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research **37**, 141–188 (2010)
24. Wallach, H.M., Murray, I., Salakhutdinov, R., Mimno, D.: Evaluation methods for topic models. In: Proceedings of the 26th annual international conference on machine learning. pp. 1105–1112 (2009)

Laplacian Centrality of Sets of Vertices of Graphs

Cristina Maier[1] and Dan A. Simovici[1]

University of Massachusetts Boston, Boston, MA, USA[1]
{cmaier,dsim}@cs.umb.edu

Abstract. Vertex centrality has important applications in fields that deal with data that can be modeled as graphs: social networks, computer networks, and biological networks to name a few. We define the Laplacian centrality of a set of vertices as a measurement of the aggregate influence that these vertices have on the connectivity of the graph.
We characterize sets of vertices whose removal have maximal effect on graph connectivity and we propose an Apriori-based algorithm for finding these sets. Experimental results show that the proposed algorithm provides a significant reduction in the search space and the execution time.

Keywords: graph Laplacian · algebraic connectivity · vertex centrality · set of vertices centrality · Apriori-based algorithm

1 Introduction

A large amount of research has been directed at identifying the most important vertices within a graph/network using various centrality measures [18]. Some of the most widely used measures of centrality are *degree centrality*, *betweenness centrality* [8], closeness centrality [3], and *eigenvector centrality*. *Katz centrality* [10] is a variant of eigenvector centrality and *PageRank* [4] is a variant of Katz centrality.

In this paper, we start from a centrality measure which evaluates vertices according to the impact their removal has on the connectedness of the graph. Let $G = (V, E)$ be a connected, undirected graph, without self loops and multiple edges. The Laplacian matrix, $L(G)$, of G is given by $L(G) = D(G) - A(G)$, where $A(G)$ is the adjacency matrix and $D(G)$ is the degree matrix of G. The algebraic connectivity of G, $\alpha(G)$ (also called *Fiedler value*) is the second smallest eigenvalue of the Laplacian matrix. A vector corresponding to the Fiedler value is called a *Fiedler vector*.

The *Laplacian centrality* of a vertex v is given by $C_L(v) = \alpha(G) - \alpha(G_{-v})$, where G_{-v} is the the graph resulted after vertex v and its incident edges are removed from graph G. The larger $C_L(v)$, the greater contribution vertex v has to connectivity of G. The effect of removing a node has on the algebraic connectivity of a graph has been studied in [11, 14, 13]. In [11] the author uses

the difference, $\phi(v) = \alpha(G) - \alpha(G\backslash v)$, and the quotient, $\kappa(v) = \dfrac{\alpha(G)}{\alpha(G\backslash v)}$, of the algebraic connectivity of the graph and the algebraic connectivity of the graph after removing a single vertex, as measures of centrality, and some bounds on ϕ and κ are provided. We refer to $\phi(v)$ as the *Laplacian centrality* of v.

Centrality measures can also be defined for sets of vertices. In [5], authors talk about degree, closeness, betweenness and flow betweenness centrality measures for groups of vertices. In [16, 17], authors introduce the notion of saturated betweenness centrality sets, and provide an algorithm for finding such sets.

In this work we propose the Laplacian centrality of a set of vertices, which measures the joint impact that removal of a set of vertices has on the connectivity of the graph. It is shown that the removal of a set of vertices can increase, decrease or have no effect on the algebraic connectivity of the graph. In some cases it can disconnect the graph. In [12] the notion of hinge is defined as a set of vertices that when removed, disconnects the graph. Our goal is to find sets of vertices that cause a maximal possible Laplacian centrality decrease.

If the intended effect on the graph is to decrease its connectedness by k, the elimination of a set of k vertices does not automatically cause this decrease. We identify those sets of vertices for which such decreases occur and we show that there are graphs for which such a decrease is impossible.

Our work is relevant for survivability studies of networks [14, 13] where authors use the Shared Backup Path Protection(SBPP) spare capacity allocation scheme to compare the impact of the algebraic connectivity and the impact of average nodal degree on the capacity allocation in network design. They examine the variation of the algebraic connectivity when removing a single vertex or a single link and conclude that the algebraic connectivity is a better indicator of network topology with regards to spare capacity allocation. Our approach allows the extension of these results to sets of vertices.

Grouping vertices together has applicability in other areas as well. Examples include detecting communities that exhibit certain properties within a social network and detecting sets of neurons that can greatly disturb the connectedness of a neuronal network.

In Section 2, we introduce Laplacian centrality of sets of vertices and we prove results relevant to Strong Laplacian Centrality (SLC) sets. In Section 3, we propose a novel Apriori-based algorithm to detect these sets. Section 4 presents experimental results. Section 5 presents final remarks and discusses future work.

2 Laplacian centrality of sets of vertices

We begin by extending the notion of vertex centrality to centrality of sets of vertices. The centrality of a set of vertices reflects the importance of sets of vertices rather than of individual vertices and is defined by the variation of algebraic connectivity when a set of vertices is removed from the graph.

Definition 1. *Let $G = (V, E)$ be a connected, undirected graph. The* Laplacian centrality *of a set U of vertices is given by*

$$C_L(U) = \alpha(G) - \alpha(G_{-U}),$$

where G_{-U} is the graph obtained by removing the vertices U and their adjacent edges from G, $\alpha(G)$ is the algebraic connectivity of graph G, and $\alpha(G_{-U})$ is the algebraic connectivity of graph G_{-U}.

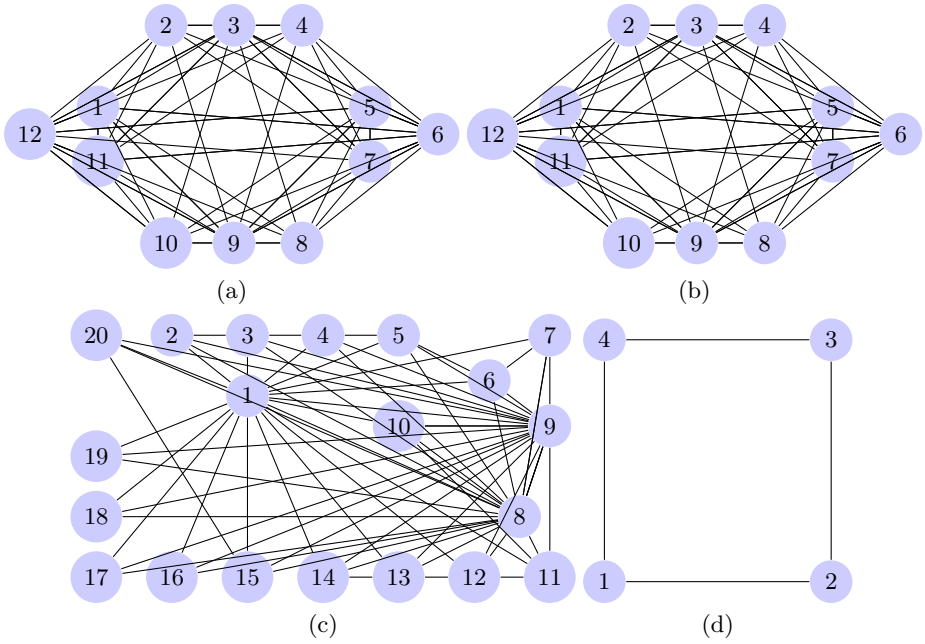


Fig. 1. Examples of Graphs

Example 1. For the graph G in Figure 1(d) all vertices have equal Laplacian centrality, namely $C_L(\{1\}) = C_L(\{2\}) = C_L(\{3\}) = C_L(\{4\}) = 1$.

The situation is different for sets of two vertices. Note that for the sets $\{1, 3\}$ and $\{2, 4\}$ we have $C_L(\{1, 3\}) = C_L(\{2, 4\}) = 2$, while $C_L(\{1, 2\}) = C_L(\{3, 4\}) = 0$. Actually, if either set $\{1, 3\}$ or $\{2, 4\}$ is removed, the graph gets disconnected.

If the set of vertices U is removed from a graph $G = (V, E)$ and $|U| = k$, then $\alpha(G_{-U}) \geqslant \alpha(G) - k$, as shown in [6, 7]. Since the remaining graph G_{-U} has $n - k$ vertices, we have $0 \leqslant \alpha(G_{-U}) \leqslant n - k$. In other words, by removing k vertices we can reduce the algebraic connectivity by at most k.

The $n \times n$ diagonal matrix having $a_1, \ldots, a_n$ on its diagonal is denoted by $\mathsf{diag}(a_1, \ldots, a_n)$. The identity matrix is $I_n = \mathsf{diag}(1, 1, \ldots, 1) \in \mathbb{R}^{n \times n}$. Also, we use the notations $\mathbf{1}_n$ and $\mathbf{0}_n$ for the n-dimensional vectors $\mathbf{1}_n = \big(1 \ldots 1\big)^T$ and $\mathbf{0}_n = \big(0 \ldots 0\big)^T$.

Definition 2. *A set of vertices U has* Strong Laplacian Centrality *(SLC) if its removal from the graph results in a decrease of the algebraic connectivity by $|U|$.*

A result of Fiedler [6, 7] is given next.

Theorem 1. *Let $G = (V, E)$ be a graph. When removing k vertices connected to all other vertices the algebraic connectivity is reduced by k.*

Proof: Let G_{-U} be the graph that is obtained after removing set of vertices U, containing vertices that are connected to all other vertices of G. To prove this we use induction on $k = |U|$; only the base step, $k = 1$ is necessary.

For the base step, $k = 1$, let $U = \{x\}$. Without loss of generality we may assume that in $L(G)$ the last column and last row correspond to vertex x. The Laplacian of G can be written as

$$L(G) = \begin{pmatrix} L(G_{-\{x\}}) + I_{n-1} & -\mathbf{1}_{n-1} \\ -\mathbf{1}_{n-1}^T & n-1 \end{pmatrix}.$$

Let $\mathbf{v}'$ be a normalized eigenvector corresponding to the algebraic connectivity α' of $G_{-\{x\}}$. We have $L(G')\mathbf{v}' = \alpha'\mathbf{v}'$. For $\mathbf{v} = \begin{pmatrix} \mathbf{v}' \\ 0 \end{pmatrix}$ we show that $L(G)\mathbf{v} = (\alpha'+1)\mathbf{v}$. Since $\mathbf{v}'$ is an eigenvalue for $G_{-\{x\}}$ corresponding to its second smallest eigenvalue, it follows that $\mathbf{1}_{n-1}^T\mathbf{v}' = 0$. This allows us to deduce

$$L(G)\mathbf{v} = L(G)\begin{pmatrix} \mathbf{v}' \\ 0 \end{pmatrix} = \begin{pmatrix} (L(G_{-\{x\}}) + I_{n-1})\mathbf{v}' \\ 0 \end{pmatrix} = (\alpha' + 1)\mathbf{v}.$$

Thus, when removing a vertex that has edges to all other vertices the algebraic connectivity decreases by 1. □

The reverse of the Theorem 1 is not necessarily true. If the algebraic connectivity decreases by 1 when removing a vertex from a graph, it does not necessarily follow that the removed vertex was connected to all other vertices as we observed in Example 1.

Corollary 1. *For a complete graph $G(V, E)$ (i.e. all vertices are fully connected) the algebraic connectivity of G equals $|V|$.*

Definition 3. *The* neighborhood of a vertex v, $N(v)$, *is the set of vertices of graph G adjacent to v.*

Definition 4. *Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$. The* support *of a vector $\boldsymbol{y} \in \mathbb{R}^n$ is the set of vertices $\mathsf{supp}(\boldsymbol{y}) = \{v_i \in V \mid y_i \neq 0\}$.*

A result of Kirkland [11] shows that if $G = (V, E)$ is a connected graph that has at least three vertices, then $\alpha(G) - \alpha(G_{-v}) = 1$ if and only if there is a Fiedler vector $\mathbf{y}$ of G such that $\mathsf{supp}(\mathbf{y}) \subseteq N(v)$.

In next theorem we propose, we extend this result to sets of vertices. The proof is inspired by the techniques developed in [6, 7, 11]. The interesting effect of this result is that even if removing each vertex of a set decreases the connectivity of the graph by 1, the removal of all vertices of this set of cardinality k may decrease the connectivity by less than k, as we saw in Example 1.

Theorem 2. *Let $G = (V, E)$ be an undirected, connected graph and let $U \subset V$ be a subset of its vertices, where $|V| = n$, $|U| = k$, $k \leqslant n - 2$ and $n \geqslant 3$. We have $\alpha(G_{-U}) = \alpha(G) - k$ if and only if there is a Fiedler vector $\boldsymbol{y}$ such that $\mathsf{supp}(\boldsymbol{y}) \subseteq \bigcap_{v_i \in U} N(v_i)$.*

Proof: Without loss of generality we assume that U consists of the last k vertices of the graph, that is, $U = \{v_{n-k+1}, \ldots, v_n\}$. Let R be the set that consists of m vertices $v \in V - U$ in the set $\bigcap_{i=n-k+1}^{n} N(v_i)$, that is, of vertices that are connected to every vertex in U. Define also the set $P = V - R - U$ consisting of $n - m - k$ vertices, which represent the vertices that do not have edges to all elements in U. Without loss of generality we may assume that $R = \{v_{n-k-m+1}, \ldots, v_{n-k}\}$, and $P = \{v_1, \ldots, v_{n-k-m}\}$. Note that if all vertices in U have edges to all other vertices in $V - U$, then $P = \emptyset$.

We begin by showing that if $\alpha(G_{-U}) = \alpha(G) - k$, then there is a Fiedler vector $\mathbf{z}$ such that $\mathsf{supp}(\mathbf{z}) \subseteq \bigcap_{i=n-k+1}^{n} N(v_i)$.

The symmetry of the Laplacian matrix $L(G)$ allows us to partition this matrix in several submatrices corresponding to the three sets of vertices, P, R, and U:

$$L(G) = \begin{pmatrix} L_P & L_{PR} & L_{PU} \\ L_{PR}^T & L_R & L_{RU} \\ L_{PU}^T & L_{RU}^T & L_U, \end{pmatrix}$$

The submatrix $L_P \in \mathbb{R}^{(n-m-k)\times(n-m-k)}$ corresponds to vertices in P . Define the matrix $L_{11} = L_P - \sum_{i=n-k+1}^{n} D_i'$. Each matrix $D_i' \in \mathbb{R}^{(n-m-k)\times(n-m-k)}$ has the form $D_i' = \mathsf{diag}(-a_{i\,1}, \ldots, -a_{i\,n-m-k})$, where a_{ij}, for $n - k + 1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n - m - m$, are entries of the Laplacian matrix $L(G)$ at positions $[i, j]$. In other words, each D_i' has on its diagonal the value node i in U adds to the degree of each node in P.

Note that not all D_i' for $n - k + 1 \leqslant i \leqslant n$ can have value 1 on same position because the vertices of P are not connected to all vertices of U.

The matrix $L_R \in \mathbb{R}^{m\times m}$ corresponds to vertices from R. Define L_{22} as

$$L_{22} = L_R - kI_m,$$

where $k = |U|$. The matrix I_m accounts for the fact that all nodes in U have edges to all nodes in R.

The matrix L_U corresponds to vertices in U. It has the degrees of the vertices of U on its main diagonal, $d_{v_{n-k+1}}, \ldots, d_{v_n}$.

The matrix $L_{PU} \in \mathbb{R}^{(n-m-k)\times k}$ can be written as

$$L_{PU} = (-\mathbf{x}_{n-k+1} \ \cdots \ -\mathbf{x}_n).$$

Observe that not all column vectors $\mathbf{x}_i$, where $n-k+1 \leqslant i \leqslant n$, can be equal to $\mathbf{1}_{n-k+1}$.

The submatrix $L_{RU} \in \mathbb{R}^{m\times k}$ can be written as $L_{RU} = (-\mathbf{1}_m, \ldots, -\mathbf{1}_m)$. Finally, we have $L_{PR} \in \mathbb{R}^{(n-m-k)\times m}$.

These observation allow us to write $L(G)$ as

$$L(G) = \begin{pmatrix} L_{11} + \sum_{i=n-k+1}^{n} D'_i & L_{PR} & L_{PU} \\ L_{PR}^T & L_{22} + kI_m & L_{RU} \\ L_{PU}^T & L_{RU}^T & L_U \end{pmatrix}.$$

After removing vertices of U from G, we get graph G_{-U}. Its Laplacian matrix, $L(G_{-U})$ is given by:

$$L(G_{-U}) = \begin{pmatrix} L_{11} & L_{PR} \\ L_{PR}^T & L_{22} \end{pmatrix}$$

Let $\mathbf{w}$ be a normalized Fiedler vector of $L(G_{-U})$. We partition this vector as $\mathbf{w} = \begin{pmatrix} \mathbf{p} \\ \mathbf{r} \end{pmatrix}$, where $\mathbf{p}$ contains the components of first $n-m-k$ elements, and $\mathbf{r}$ contains the last m components of $\mathbf{w}$. Let $\mathbf{z}$ be the vector of length n given by $\mathbf{z} = \begin{pmatrix} \mathbf{p} \\ \mathbf{r} \\ \mathbf{0}_k \end{pmatrix}$, where $\mathbf{0}_k$ is the zero vector of length k.

Note that we have $\mathbf{w}^T\mathbf{w} = 1$ and $\mathbf{w}^T\mathbf{1}_{n-k} = \mathbf{1}_{n-k}^T\mathbf{w} = 0$. This amounts to $\mathbf{p}\mathbf{1}_{n-m-k}^T + \mathbf{r}\mathbf{1}_m^T = 0$. Also, we have $\mathbf{p}^T\mathbf{p} + \mathbf{r}^T\mathbf{r} = 1$. Therefore, $\mathbf{z}^T\mathbf{z} = 1$ and $\mathbf{z}^T\mathbf{1}_n = \mathbf{1}_n^T\mathbf{z} = 0$, which implies:

$$\mathbf{w}^T L(G_{-U})\mathbf{w} = \mathbf{p}^T L_{11}\mathbf{p} + \mathbf{r}^T L_{PR}^T\mathbf{p} + \mathbf{p}^T L_{PR}\mathbf{r} + \mathbf{r}^T L_{22}\mathbf{r},$$

and

$$\begin{aligned} \mathbf{z}^T L(G)\mathbf{z} = \mathbf{p}^T L_{11}\mathbf{p} + &\sum_{i=n-k+1}^{n} \mathbf{p}^T D'_i\mathbf{p} \\ &+\mathbf{r}^T L_{PR}^T\mathbf{p} + \mathbf{p}^T L_{PR}\mathbf{r} + \mathbf{r}^T L_{22}\mathbf{r} + \mathbf{r}^T kI_m\mathbf{r}. \end{aligned}$$

The previous equalities yield

$$\mathbf{z}^T L(G)\mathbf{z} = \mathbf{w}^T L(G_{-U})\mathbf{w} + \sum_{i=n-k+1}^{n} \mathbf{p}^T D'_i\mathbf{p} + k\mathbf{r}^T I_m\mathbf{r}. \tag{1}$$

We have:

$$\alpha(G_{-U})\mathbf{z}^T\mathbf{z} + k\mathbf{p}^T\mathbf{p} + k\mathbf{r}^T\mathbf{r} = (\alpha(G_{-U}) + k)\mathbf{z}^T\mathbf{z} = \alpha(G)\mathbf{z}^T\mathbf{z} \leqslant \mathbf{z}^T L(G)\mathbf{z}. \tag{2}$$

Here we used $\alpha(G_{-U}) + k = \alpha(G)$.

Courant-Fischer's theorem implies that $\alpha(G_{-U}) \leq \mathbf{t}^T L(G_{-U})\mathbf{t}$ for every $\mathbf{t}$ such that $\mathbf{t}^T \mathbf{1}_{n-k} = 0$ and $\mathbf{t}^T\mathbf{t} = 1$. Therefore, we have $\alpha(G)\mathbf{z}^T\mathbf{z} \leqslant \mathbf{z}^T L(G)\mathbf{z}$.

Combining Equalities (1) with (2), and using the fact that

$$\mathbf{w}^T L(G_{-U})\mathbf{w} = \mathbf{w}^T \alpha(G_{-U})\mathbf{w} = \alpha(G_{-U})\mathbf{w}^T\mathbf{w} = \alpha(G_{-U}) = \alpha(G_{-U})\mathbf{z}^T\mathbf{z}$$

we get:

$$\begin{aligned}\alpha(G_{-U})\mathbf{z}^T\mathbf{z} + k\mathbf{p}^T\mathbf{p} + k\mathbf{r}^T\mathbf{r} &\leqslant \mathbf{w}^T L(G_{-U})\mathbf{w} + \sum_{i=n-k+1}^{n} \mathbf{p}^T D'_i\mathbf{p} + k\mathbf{r}^T I_m\mathbf{r} \\ &= \alpha(G_{-U})\mathbf{z}^T\mathbf{z} + \sum_{i=n-k+1}^{n} \mathbf{p}^T D'_i\mathbf{p} + k\mathbf{r}^T I_m\mathbf{r}.\end{aligned}$$

This can be rewritten as:

$$\alpha(G_{-U})\mathbf{z}^T\mathbf{z} + k\mathbf{p}^T\mathbf{p} + k\mathbf{r}^T\mathbf{r} \leqslant \alpha(G_{-U})\mathbf{z}^T\mathbf{z} + \sum_{i=n-k+1}^{n} \mathbf{p}^T D'_i\mathbf{p} + k\mathbf{r}^T\mathbf{r}. \quad (3)$$

Note that

$$\begin{aligned}\mathbf{p}^T D'_{n-k+1}\mathbf{p} &< \mathbf{p}^T I_{n-m-k}\mathbf{p} = \mathbf{p}^T\mathbf{p} \\ &\vdots \\ \mathbf{p}^T D'_n\mathbf{p} &< \mathbf{p}^T I_{n-m-k}\mathbf{p} = \mathbf{p}^T\mathbf{p},\end{aligned}$$

because the diagonal matrices D'_i, for $n-k+1 \leqslant i \leqslant n$ cannot all have 1 on the same position on the diagonal. From (3) applying the previous inequalities it follows that $\mathbf{p}$ must be $\mathbf{0}_{n-m-k}$. We will show that this also means that $\mathbf{z}$ must be a Fiedler vector for G.

Because $\mathbf{p}$ is $\mathbf{0}_{n-m-k}$, we have: $\mathbf{w} = \begin{pmatrix}\mathbf{0}_{n-m-k}\\ \mathbf{r}\end{pmatrix}$, and $\mathbf{z} = \begin{pmatrix}\mathbf{0}_{n-m-k}\\ \mathbf{r}\\ \mathbf{0}_k\end{pmatrix}$. Since $\mathbf{w}$ is a Fiedler vector for $L(G_{-U})$, we have

$$Ł(G_{-U})\mathbf{w} = \begin{pmatrix}L_{11}\mathbf{p} + L_{PR}\mathbf{r}\\ L_{PR}^T\mathbf{p} + L_{22}\mathbf{r}\end{pmatrix} = \begin{pmatrix}L_{PR}\mathbf{r}\\ L_{22}\mathbf{r}\end{pmatrix} = \alpha(G_{-U})\begin{pmatrix}\mathbf{0}_{n-m-k}\\ \mathbf{r}\end{pmatrix}$$

because $\mathbf{p}$ is $\mathbf{0}_{n-m-k}$. This means that we have $L_{PR}\mathbf{r} = \mathbf{0}_{n-m-k}$ and $L_{22}\mathbf{r} = \alpha(G_{-U})\mathbf{r}$. We also have:

$$Ł(G)\mathbf{z} = \begin{pmatrix}L_{PR}\mathbf{r}\\ L_{22}\mathbf{r} + k\mathbf{r}\\ \mathbf{0}_k\end{pmatrix} = \begin{pmatrix}\mathbf{0}_{n-m-k}\\ \alpha(G_{-U})\mathbf{r} + k\mathbf{r}\\ \mathbf{0}_k\end{pmatrix} = \alpha(G_{-U} + k)\mathbf{z} = \alpha(G)\mathbf{z}.$$

Thus, $\mathbf{z}$ is a Fiedler vector of G such that $\mathsf{supp}(\mathbf{z}) \subseteq \bigcap_{i=n-k+1}^{n} N(v_i)$.

Conversely, let $G = (V, E)$ be a graph for which U is a set of k vertices, $U = \{v_{n-k+1}, \ldots, v_n\}$, such that G has a Fiedler vector $\mathbf{y}$ with $\mathsf{supp}(\mathbf{y}) \subseteq \bigcap_{i=n-k+1}^{n} N(v_i)$. We show that $\alpha(G) - \alpha(G_{-U}) = k$.

The Laplacian matrix of G can be partitioned in the following submatrices corresponding to two sets of vertices, U and $V-U$:

$$L(G) = \begin{pmatrix} L_{V-U} & L_{(V-U)U} \\ L^T_{(V-U)U} & L_U \end{pmatrix},$$

where L_{V-U} is the $(n-k)\times(n-k)$ submatrix corresponding to vertices from $V-U$. We have $L(G_{-U}) = L_{V-U} - \sum_{i=n-k+1}^{n} D_i$, where each matrix $D_i \in \mathbb{R}^{(n-k)\times(n-k)}$ has the form $D_i = \mathsf{diag}(-a_{i1},\ldots,-a_{i\,n-k})$ where a_{ij} are entries of the Laplacian matrix $L(G)$, and $n-k+1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n-k$.

The submatrix $L_U \in \mathbb{R}^{k\times k}$ corresponds to vertices of U. We have $L_{(V-U)U} \in \mathbb{R}^{(n-k)\times k}$. Now, the Laplacian matrix of G can be written as:

$$L(G) = \begin{pmatrix} L(G_{-U}) + \sum_{i=n-k+1}^{n} D_i & L_{(V-U)U} \\ L^T_{(V-U)U} & L_U \end{pmatrix}.$$

We assume that G has a Fiedler vector $\mathbf{y}$ whose support is a subset of $N(v_{n-k+1}) \cap N(v_{n-k+2}) \cap \cdots \cap N(v_n)$, which can be written as: $\mathbf{y} = \begin{pmatrix} \tilde{\mathbf{y}} \\ \mathbf{0}_k \end{pmatrix}$, where $\tilde{\mathbf{y}}$ is a vector of size $n-k$.

We have $\mathbf{y}^T\mathbf{y} = 1$ (because $\mathbf{y}$ is normalized) and $\mathbf{y}^T\mathbf{1}_n = \mathbf{1}_n^T\mathbf{y} = 0$, which implies $\tilde{\mathbf{y}}^T\tilde{\mathbf{y}} = 1$ and $\tilde{\mathbf{y}}^T\mathbf{1}_{n-k} = \mathbf{1}_{n-k}^T\tilde{\mathbf{y}} = 0$.

The definition of the Fiedler vector implies:

$$\mathbf{y}^T L(G)\mathbf{y} = \mathbf{y}^T \alpha(G)\mathbf{y} = \alpha(G). \tag{4}$$

Additionally, we have:

$$\mathbf{y}^T L(G)\mathbf{y} = \tilde{\mathbf{y}}^T L(G_{-U})\tilde{\mathbf{y}} + \sum_{i=n-k+1}^{n} \tilde{\mathbf{y}}^T D_i \tilde{\mathbf{y}} \tag{5}$$

Equalities (4) and (5) show that

$$\tilde{\mathbf{y}}^T L(G_{-U})\tilde{\mathbf{y}} = \alpha(G) - \sum_{i=n-k+1}^{n} \tilde{\mathbf{y}}^T D_i \tilde{\mathbf{y}}.$$

Note that we have $\tilde{\mathbf{y}}^T D_i \tilde{\mathbf{y}} = 1$, for all $n-k+1 \leqslant i \leqslant n$, because $\tilde{\mathbf{y}}^T\tilde{\mathbf{y}} = 1$, and because from the hypothesis we know that the support of $\mathbf{y}$ (and therefore the support of $\tilde{\mathbf{y}}$) is a subset of $N(v_{n-k+1}) \cap N(v_{n-k+2}) \cap \ldots \cap N(v_n)$. Therefore,

$$\tilde{\mathbf{y}}^T L(G_{-U})\tilde{\mathbf{y}} = \alpha(G) - k. \tag{6}$$

Again, using the Courant-Fischer's theorem, we have

$$\alpha(G_{-U}) \leqslant \tilde{\mathbf{y}}^T L(\tilde{G_{-U}})\tilde{\mathbf{y}}. \tag{7}$$

Combining Equalities (6) and (7) we get

$$\alpha(G_{-U}) \leqslant \alpha(G) - k. \tag{8}$$

Fiedler's results [6, 7] imply

$$\alpha(G_{-U}) \geq \alpha(G) - k. \tag{9}$$

From (8) and (9) it follows that $\alpha(G_{-U}) = \alpha(G) - k$, which concludes the proof. □

Example 2. Using Theorem 2 on the same graph as in Example 1, from Figure 1(d) we can determine the SLC sets without having to calculate the algebraic connectivity of the resulting subgraphs after sets of vertices are removed. For this graph the algebraic connectivity has multiplicity 2. Therefore, the Fiedler vector space is span by two non-zero eigenvectors. We have Fiedler vectors $\mathbf{f}_1^T = [0, -0.7071068, 0, 0.7071068]$ and $\mathbf{f}_2^T = [0.7071068, 0, -0.7071068, 0]$. Neighborhoods of the vertices from our graph are: $N(1) = \{2, 4\}, N(2) = \{1, 3\}$, $N(3) = \{2, 4\}, N(4) = \{1, 3\}$. We see that $\mathbf{f}_1$ has support in $N(1)$, $N(3)$, and $N(1) \cap N(3)$; $\mathbf{f}_2$ has support in $N(2)$, $N(4)$, and $N(2) \cap N(4)$. It follows that $\{1\}, \{2\}, \{3\}, \{4\}, \{1, 3\}$, and$\{2, 4\}$ have Strong Laplacian Centrality. There is no Fiedler vector that has support in $N(1) \cap N(2)$, and $N(3) \cap N(4)$, which means sets $\{1, 2\}$ and $\{3, 4\}$ are not SLC sets.

Next, we introduce a theorem that plays a crucial role in the algorithm from Section 3.

Theorem 3. *Let $G = (V, E)$ be an undirected, connected graph and let $U \subset V$ be a subset of its vertices, where $|V| = n$, $|U| = k$, $k \leqslant n - 2$ and $n \geqslant 3$. If $\alpha(G_{-U}) = \alpha(G) - k$, then for any subset U' of U with $|U'| = \ell$, $1 \leqslant l \leqslant k$ we have $\alpha(G_{-U'}) = \alpha(G) - \ell$.*

Proof: Let G be the original graph, with algebraic connectivity $\alpha(G)$ and let U be a set of vertices such that $|U| = k$ and $\alpha(G_{-U}) = \alpha(G) - k$.

Let U' be a subset of U such that $|U'| = \ell$, where $1 \leqslant \ell \leqslant k$. A result in [7] implies

$$\alpha(G_{-U'}) \geqslant \alpha(G) - \ell. \tag{10}$$

Since G_{-U} can be obtained by removing the set of vertices $U \setminus U'$(i.e. set U minus sets U') from graph $G_{-U'}$, we have:

$$\alpha(G_{-U}) \geqslant \alpha(G_{-U'}) - (k - \ell) \geqslant \alpha(G) - \ell - (k - \ell) = \alpha(G) - k. \tag{11}$$

Therefore, $\alpha(G_{-U'}) = \alpha(G) - \ell$, which concludes the proof. □

The reverse of Theorem 3 is not necessarily true. As we saw in Example 1, we can have a set of k vertices that individually reduce α by 1, but when removed together they do not reduce α by k.

3 An Apriori-based algorithm to determine SLC sets.

We seek to determine all sets of vertices that have Strong Laplacian Centrality. In other words, if $G = (V, E)$ is a graph with n nodes, $n \geqslant 3$, we seek to determine all sets $U \in V$, with $|U| \leqslant n - 2$ such that $\alpha(G) - \alpha(G_{-U}) = |U|$.

One way to compute these sets is to examine all possible combinations of vertices and determine if they satisfy the above condition. We call this the *brute force algorithm*. While this could work for small graphs, it is not practical for larger graphs, which would require to examine $\sum_{i=1}^{n-2} \binom{n}{i} = 2^n - n - 2$ sets, a number that grows exponentially with the number of vertices in the graph.

To reduce the search space, an early stop can be considered on brute force algorithm by limiting the size of sets that can have Strong Laplacian Centrality. We call this the *brute force early-stop algorithm*. Since the algebraic connectivity of the graph represents an upper bound of the size of the sets that can have Strong Laplacian Centrality we could reduce the search space accordingly. However, this will not reduce the number of vertices that can be part of the candidate set. Also, the denser the graph is (i.e. the more edges it has), the less impact this bound will have.

Algorithm 3.1: ApropriBasedAlgorithm

input : The graph's adjacency matrix $A(G) \in \mathbb{R}^{n \times n}$
output: A map$< sizeSet, listSets >$ containing all sets U of vertices, $1 \leqslant |U| \leqslant k$, where $\alpha(G) - \alpha(G_{-U}) = |U|$

```
1  begin
2      n = number of rows in A(G); k = n − 2;
3      D(G) = calculate degree matrix from A(G);
4      L(G) = D(G) − A(G); // the Laplacian matrix
5      V = list of sets of each vertex from G;
       // this will hold map< sizeSet, listSets >
6      resultMap = empty map < Integer, List < Set < Integer >>>;
7      if isCompleteGraph(L(G)) then
8          return generateAllPossibleSets(V,n,k);
9      end
10     α(G) = calculateAlgebraicConnectvity(L(G));
11     i = 2;
12     okSetsSize1 = verifySLCSets(V,L(G),α(G));
13     resultMap[1] = okSetsSize1;
14     okSetsPrevSize = okSetsSize1;
15     while i ⩽ k AND okSetsPreviouseSize <> [] do
16         candidateSetsSizeI = generateCandidates(okSetsPrevSize,okSetsSize1);
17         okCandidateSetsSizeI = verifySLCSets(candidateSetsSizeI,L(G), α(G));
18         resultMap[i] = okCandidateSetsSizeI;
19         okSetsPrevSize = okCandidateSetsSizeI;
20     end
21     return resultMap;
22 end
```

The *Apriori* $[1, 2]$*-based algorithm* we propose goes a step further by limiting the number of vertices that can be members of these sets. This algorithm is

based on Theorem 3 that stipulates that if a set of vertices has Strong Laplacian Centrality, then all its subsets enjoy the same property. Using this property we significantly reduce the number of candidate sets of vertices. For example, vertex v will not be considered as a member of any candidate set of size two or higher unless the Laplacian centrality of $\{v\}$ equals 1. To further optimize the solution we use Theorem 1, which shows that when a set of vertices U is fully connected to the rest of the graph (i.e. each vertex from U has edges to all vertices of the graph), then the Laplacian centrality of U equals $|U|$. Parallelization techniques, such as examining candidate sets of same size simultaneously could further improve the performance of the algorithm.

Algorithm 3.1 presents the main pseudocode of the Apriori-based algorithm. The function verifySLCSets(listSets, L(G), $\alpha(G)$) checks each set in listSets and returns only those sets that have Strong Laplacian Centrality. For each set, it checks whether the set contains only fully connected vertices, and if not, the Laplacian centrality is calculated and checked if it is maximal. Note that verifying the elements of listSets could be done in parallel. The function generateCandidates(okSetsPrevSize,okSetsSize1) produces candidates sets of size $PrevSize+1$ by adding one element from okSetsSize1 to each element from okSetsPrevSize, removing duplicates, if any.

Observe that for any complete graph, K_n, our Apriori-based algorithm does not examine any set, as it determines that all possible combinations (of any size) of vertices have Strong Laplacian Centrality.

4 Experimental results

We present some experimental results on both artificial and real-world graphs. Our proposed Apriori-based algorithm outperforms the brute force approaches. All algorithms were implemented in R and they all use the optimization given by Theorem 1. The experiments were performed on a Mac OS computer with a 3.5 GHz Intel Core i7 processor, and 16 GB 1600 MHz DDR3 of RAM. Note that a variation of the Apriori-based algorithm using the upper limit from the brute force early-stop showed no further significant improvement. In all cases, we present the average execution time for 100 runs.

For the graph shown in Figure 1(a), the brute force algorithm has to examine 4082 sets, whereas the brute force early-stop needs to examine 1585 sets. Our Apriori-based algorithm examines only 23 sets and produces the same result. It examines 12 sets of size one, 6 sets of size two, 4 sets of size three and 1 set of size four. The result shows that the Strong Laplacian Centrality sets are: $\{3\}$, $\{6\}$, $\{9\}$,$\{12\}$, $\{3,6\}$, $\{3,9\}$, $\{3,12\}$, $\{6,9\}$, $\{6,12\}$, $\{9,12\}$, $\{3,6,9\}$, $\{3,6,12\}$, $\{3,9,12\}$, $\{6,9,12\}$, and $\{3,6,9,12\}$. The average execution time of the brute force approach is 2.720 seconds, and of brute force early-stop is 0.8958 seconds. The average execution time of the Apriori-based algorithm is 0.011164 seconds.

To determine all sets with Strong Laplacian Centrality for the graph shown in Figure 1(b), the brute force algorithm examines 4082 sets. Brute force early-stop examines 793 sets. Using our Apriori-based algorithm we only examine 13

sets and return the same result. We examine 12 sets of size one, and 1 set of size two. The result shows that the Strong Laplacian Centrality sets are: singletons $\{6\}$, $\{12\}$, and set of size two $\{6, 12\}$. The average execution time of the brute force approach is 2.717 seconds, and of brute force early-stop is 0.4138 seconds. The average execution time of the Apriori-based algorithm is 0.005286 seconds.

For the graph given in Figure 1(c), the brute force algorithm needs to examine $1,048,554$ sets. In this case, brute force early-stop provides a great improvement, having to examine only 1350 sets. Our Apriori-based solution takes this a step further and examines only 24 sets, producing the same result, which shows that the maximal Laplacian centrality sets are: the singletons $\{1\}$, $\{8\}$, $\{9\}$, the sets of size two $\{1, 8\}$, $\{1, 9\}$, $\{8, 9\}$, and one set of size three $\{1, 8, 9\}$. Note that vertices 1, 8 and 9 are fully connected. Therefore during the examination of subsets of $\{1, 8, 9\}$ the algorithms determine without further computation that every subset of $\{1, 8, 9\}$ has Strong Laplacian Centrality. While the average execution time of the brute force approach is 958.6 seconds, and of brute force early-stop is 0.7294 seconds, the average execution time of the Apriori-based algorithm is only 0.007847 seconds.

Further experiments were performed on two real-world data sets: the *Dolphins Network* and the *American College Football Network*.

The Dolphins Network [15] represents an undirected network of frequent association between 62 dolphins living in Doubtful Sound, New Zealand. The network contains 159 edges representing frequent interactions between the dolphins. We are interested in finding whether this network contains any SLC sets. Our algorithm found that no SLC sets are present in this network. The Apriori-based algorithm only needed to verify 62 sets of single vertices, and it ran in 2.765 seconds.

The American Football Network, introduced in [9], is a representation of the games of Division I, played in 2000 season. An edge between two vertices (representing two teams) means there was a game played between these teams. The network has 115 vertices connected by 613 edges. The network also incorporates a community structure. The teams are grouped into twelve conferences (i.e. communities).

Our Apriori-based algorithm finds that this network contains no SLC sets. The Apriori-based algorithm ran in 9.682 seconds, and it only had to verify 115 sets, whereas the brute force algorithm would have to verify $2^{115} - 117$ sets.

Even though we found no SLC sets in the entire network, we found some interesting results when we looked for SLC sets in each of the twelve communities. Eight of the twelve communities contain no SLC sets; the remaining four communities contain a large number of SLC sets. Out of these four communities, two of them form a complete subgraph. The other two sets contain a large number of SLC sets, 501 and 780, respectively. This suggests that these communities have teams whose removal may strongly affect the interactions between teams that belong to the community.

Table 1 contains details of these findings.

Table 1. American Football SLC sets

Conference/ Community	No. of Teams	No. of SLC Sets	Apriori-based Running Time	Comments
(0)Atlantic Coast	9	501	0.1436 s	
(1)Big East	8	2035	0.0673 s	Complete graph
(2)Big Ten	11	0	0.09746 s	
(3)Big Twelve	12	0	0.1128 s	
(4)Conference USA	10	0	0.08122 s	
(5)Independents	5	0	0.02454 s	
(6)Mid-American	13	0	0.1211 s	
(7)Mountain West	8	246	0.06564 s	Complete graph
(8)Pacific Ten	10	780	28.40 s	Verified 1012 sets
(9)Southeastern	12	0	0.1171 s	
(10)Sun Belt	7	0	0.04354 s	
(11)Western Athletic	10	0	0.08111 s	

5 Conclusions and Future Work

In this paper we introduced the Laplacian Centrality of a set of vertices, which quantifies the joint impact these vertices have on the connectedness of the graph. This impact does not necessarily amount to the sum of the impacts of each individual vertex, as we saw in Section 2.

Laplacian centrality of sets of vertices can have important applications in computer networks, social networks, and biological networks. In computer networks, Laplacian centrality might help with decisions related to spare-capacity allocation. Determining which sets of vertices have a high Laplacian centrality value, could give us an indication on which network nodes need more protection as a group (i.e. if all these nodes fail, the network's connectivity will be greatly impacted). In biology, the Laplacian centrality of a set of neurons would indicate the impact the destruction of those neurons would have on the connectedness of the network of neurons. Therefore, finding sets of neurons with high Laplacian centrality could provide valuable information. In social networks we can identify sets of persons whose removal can greatly disturb the connectivity of the network. This could provide opportunities to deploy early intervention measures to re-engage people, when multiple persons from such a particular set are leaving the social platform.

The maximal possible Laplacian centrality value of a set of k vertices is k. This means that when k vertices are removed from a graph, the algebraic connectivity can decrease by at most k. Hence, we defined the Strong Laplacian Centrality (SLC) sets as sets which have a Laplacian centrality equal to $|S|$.

In a social network we might want to determine SLC sets of a given size k, as members of such a set exhibit some strong special connectivity within the network. Or, we might want to determine communities which contain SLC sets, which would indicate some topological characteristic of the community, and could further enable us to partition the communities into multiple classes.

We proved some properties related to algebraic connectivity of graphs obtained by deleting sets of vertices. These results are useful in determining SLC sets. One of these theorems allowed us to introduce an Apriori-based algorithm that detects SLC sets more efficiently.

We presented experimental results for detecting SLC sets within three artificial, and two real-world networks. We compared the running time and search-space of the brute force, the brute force early-stop, and the Apriori-based algorithms implementations and presented the findings. These results demonstrate that the Apriori-based algorithm significantly reduces both the search space and the execution time. This matters for larger graphs, where the brute force algorithm, and even the brute force early-stop algorithm, can be prohibitively expensive. These experiments also show that there are networks in which SLC sets do not exist. The experiments that were run on American College Football Network, show that the football communities (i.e. conferences) have interesting characteristics relative to SLC sets. While some of this network communities contain no SLC sets, the communities that do, contain a large number of SLC sets.

We plan to enhance the Apriori-based algorithm by using Theorem 2 to determine SLC sets, without calculating the algebraic connectivity of subgraphs. This should result in making this algorithm scalable for larger graphs. We shall further examine if our approach to Laplacian centrality of sets of vertices can be extended to other centrality measures.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 International Conference on Management of Data*, pages 207–216, Washington, D.C., 1993. ACM, New York.
2. R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.
3. Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.
4. Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *IEEE Data Eng. Bull.*, 21(2):37–47, 1998.
5. Martin Everett and Stephen Borgatti. The centrality of groups and classes. *Journal of Mathematical Sociology*, 23:181–201, 01 1999.
6. Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.
7. Miroslav Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25(1):57–70, 1989.
8. Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.
9. Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.
10. Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

11. Steve Kirkland. Algebraic connectivity for vertex-deleted subgraphs, and a notion of vertex centrality. *Discrete Mathematics*, 310(4):911–921, 2010.
12. Steve Kirkland, Israel Rocha, and Vilmar Trevisan. Algebraic connectivity of k-connected graphs. *Czechoslovak Mathematical Journal*, 65(1):219–236, 2015.
13. William Liu, Krzysztof Pawlikowski, and Harsha Sirisena. Algebraic connectivity metric for spare capacity allocation problem in survivable networks. *Computer Communications*, 34(12):1425–1435, 2011.
14. William Liu, Harsha Sirisena, and Krzysztof Pawlikowski. Efficacy of fiedler value versus nodal degree in spare capacity allocation. In *2009 15th Asia-Pacific Conference on Communications*, pages 678–681. IEEE, 2009.
15. David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
16. Cristina Maier and Dan Simovici. Betweenness centrality of sets of vertices in graphs: which vertices to associate. *Proceedings of ICISDM - in print*, 2020.
17. Cristina Maier and Dan Simovici. Saturated betweenness centrality sets of vertices in graphs. *Journal of Advances in Information Technology (JAIT) - in print*, 2020.
18. M. Newman. *Networks*. Oxford University Press, 2018.

New Machine Learning Approaches to Improve Software Bug Prediction

Abdullah Algarni, Emad Kaen, and Turki Turki

Department of Computer Science, King Abdulaziz University,
P.O. Box 80221, Jeddah 21589, Saudi Arabia
{amsalgarni,tturki}@kau.edu.sa

Abstract. Predicting bugs of a software system is an important research problem in software engineering. By predicting software bugs correctly, developers can accelerate the testing process for locating source code components containing bugs and hence reduce the time associated with software maintenance and development, thereby improving the software development cycle. In today's software industry, plethora of software engineering environments has led to substantial amounts of data stored in repositories. Mining such data is a challenging task. A key goal of this study is to deliver reliable machine learning tools to software developers, who would use these tools as prediction calculators to identify bugs in software systems. To accomplish this goal, we develop new machine learning approaches combining an unsupervised technique with or without feature selection and supervised learning techniques. Experimental results on various bug prediction datasets demonstrate that our machine learning approaches generate higher performance results when compared against existing baseline approaches.

Keywords: Machine learning, Supervised and unsupervised learning, Software metrics, Bug prediction, Mining software repositories

1 Introduction and Related Work

Software bugs have a significant impact on the software development process; they are one of the leading causes to delay software projects [9]. Software developers act differently when dealing with software bugs due to several factors including coding experience, changing of software requirements, losing key players, and lacking existing tools to identify bugs. The increasing number of delayed software projects caused by software bugs is critical for software managers and has attracted machine learning researchers and others including software engineers to tackle this problem affecting the software industry [22, 23, 3, 28, 2, 24, 31, 27, 36, 15, 12, 26, 42, 21, 30, 32].

Correctly predicting bugs in advance leads to fast human inspection of codes as well as localization of bugs in given source code files [20]. One of the earliest attempts to tackle the bug prediction problem was made by Basili et al. [1], who utilized logistic regreassion coupled with the Chidamber and Kemerer (CK) metrics, and assessed their usefulness in bug prediction. Nagappan et al. [25] exploited the CK metrics as well as other complexity measures, which were provided to a machine learning algorithm for software bug prediction. The CK metrics have also

been used by other researchers to improve the performance of bug prediction [36, 10, 41]. Nucci et al. [9] proposed scattering metrics as features to enhance the prediction performance.

Based on the history of antipatterns in files, Taba et al. [37] proposed four metrics, namely Average Number of Antipatterns (ANA), Antipattern Complexity Metric (ACM), Antipattern Recurrence Length (ARL), and Antipattern Cumulative Pairwise Differences (ACPD). Experimental results on two real datasets demonstrated the usefulness of these four metrics for bug prediction.

Jureczko et al. [19] proposed a machine learning approach incorporating twenty structural metrics (i.e., features), provided to a machine learning algorithm to learn a model. The generated model was then used to predict buggy instances. Shivaji et al. [35] proposed a machine learning approach coupled with feature selection techniques, to improve the bug prediction performance.

Palomba et al. [29] proposed a feature-based machine learning approach to software bug prediction, which worked as follows. A preprocessing step including a feature selection procedure was applied to an initial dataset, to yield a dataset with selected features. Then, they concatenated basic metrics (features) and code smell intensity. The resulted dataset was provided to a machine learning algorithm, inducing a model, which was subsequently applied to unseen examples to generate predictions. Experimental results demonstrated the discriminative power of their approach.

Although the previous approaches aim to include software metrics to improve software bug prediction, the performance results of these approaches are far from being perfect. There is much room to improve the use of the software metrics and machine learning algorithms.

The main contributions of this work are the following:

1. We present two new machine learning approaches for software bug prediction. The first machine learning approach (FML) aims to select important metrics (i.e., features) using an ensemble method coupled with decision trees. The second machine learning approach is an extension of the FML utilizing an unsupervised learning algorithm, i.e., K-means, to improve the extraction process of features [39].
2. By processing data using our approaches, we extracted important software metrics (i.e., features), to be used as markers guiding the classification process.
3. We employed supervised learning algorithms coupled with our approaches, to assess the applicability of a subset of software metrics to predict software bugs.
4. Experimental results on several datasets pertaining to Software Apache Ant and Software Apache Camel demonstrate that our approaches outperform prediction algorithms employing several baselines utilizing different sets of software metrics including those of Palomba et al. [19, 29, 37].

The main observations of this work are the following:

1. All proposed approaches indicate that not all software metrics are important in guiding the classification process for bug prediction.
2. Both of the two machine learning approaches we present here indicate that the following software metrics are important metrics to improve the bug prediction process when coupled with machine learning algorithms: Feature Introduction Changes (FI.CHANGES), Scattering Metric (SCATTERING), and Intensity of Code Smells (INTENSITY).
3. These selected metrics can be used by machine learning and software engineering researchers as important bug predictors (i.e., features).

The rest of this paper is organized as follows. Section 2 presents our machine learning approaches. Section 3 reports results on several datasets pertaining to software apache projects.

Section 4 discusses and explains the results. Section 5 concludes the paper and points out some directions for future work.

2 Our Proposed Approaches

2.1 The First Machine Learning Approach (FML)

Figure 1 illustrates the first machine learning approach (FML), which works as follows. Suppose that we are given a dataset divided into training set $\{(x_i, y_i)\}_{i=1}^{m}$ and a test set $\{x_i^{'}\}_{i=1}^{n}$ which consists of *m* and *n* instances (i.e., examples or code components), respectively. In the training set, $x_i \in R^u$ is the *i*th training example, *u* is the number of features, and $y_i \in \{\text{buggy}, \text{non-buggy}\}$ is the label of x_i. In the target test set, $x_i^{'} \in R^u$ is the *i*th testing example of *u* features. In (I), the training set is provided as an input to five fitted decision trees (i.e., 5 models), where each node in a model corresponds to a feature. The output from (II) corresponds to extracted features, where we take the union of all features (see III). In (IV), a training set with selected features from III is provided as an input to a machine learning algorithm, to induce a model *h*. In (V), we select the same features of the test set as those selected in (IV). Then, model *h* is applied to the test set of selected features, to generate predictions corresponding to buggy or non-buggy (i.e., clean) components.

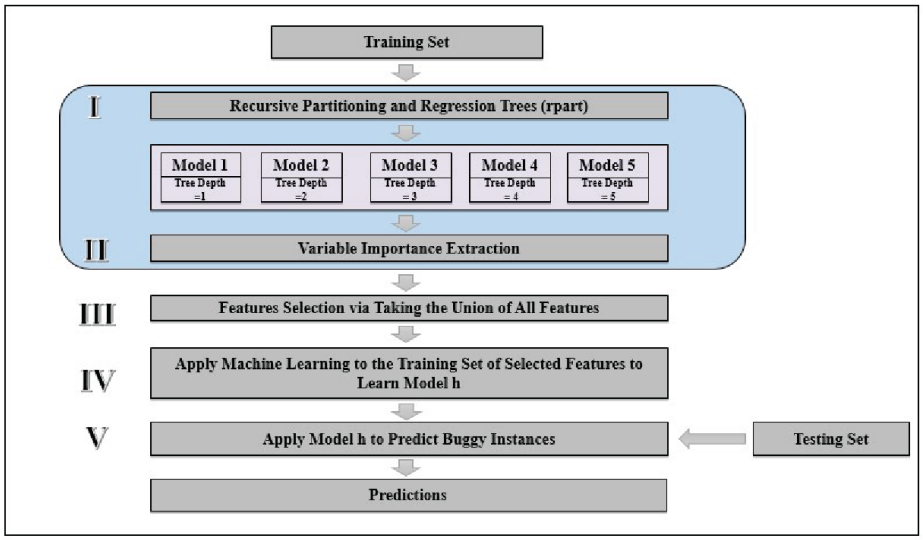


Figure 1: The first machine learning approach, FML, for predicting buggy instances.

2.2 The Second Machine Learning Approach (SML)

Figure 2 shows the second machine learning approach (SML). The SML is like the FML. The only difference is that the SML includes additional steps (see (I), (II) and (III)), where unlabeled training and test sets are combined and then provided to the K-means algorithm, which is applied to cluster data into two clusters [17]. For each cluster (see (III)), we assign training examples and their corresponding labels (obtained from the training set) to the next step. The remaining steps (i.e., (IV), (V), (VI), (VII), and (VIII)) are the same as the steps (i.e., (I), (II), (III), (IV) and (V)) in FML.

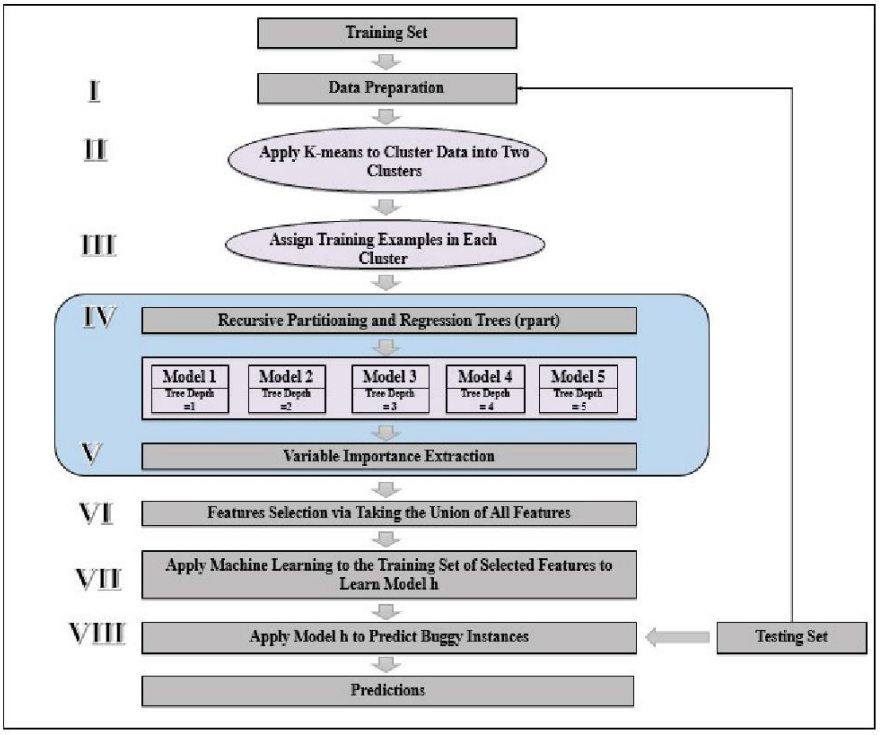


Figure 2: The Second machine learning approach, SML, for predicting buggy instances.

3 Experiments and Results

We conducted an experimental study to report performance results of the proposed approaches as well as prediction algorithms employing several baseline approaches. This section first describes datasets pertaining to software projects for the task of bug prediction, then details the experimental methodology we used, and finally reports experimental results.

3.1 Datasets

Table 1 provides statistics about the number of instances and distribution of class labels of each dataset used in this study [29, 36].

Table 1: Details of our software project datasets.

System Name	Releases	Instances	Buggy Instances	Non-Buggy Instances
Apache Ant	Ant-1.3	125	20	105
	Ant-1.4	178	40	138
	Ant-1.5	293	32	261
	Ant-1.6	351	92	259
	Ant-1.7	745	166	579
Apache Camel	Camel-1.0	339	13	326
	Camel-1.2	608	216	392
	Camel-1.4	872	145	727
	Camel-1.6	965	188	777

Table 2 lists all Chidamber and Kemerer (CK) metrics used in Software Apache Ant and Software Apache Camel. The table describes 20 structural metrics as well as 6 CK metrics.

Table 2: Summary of structural metrics and Chidamber and Kemerer (CK) metrics (shown in bold) for the datasets of each software project.

Abbrev	Feature	Type of Feature
WMC	**Weighted Methods per Class**	**Numerical**
DIT	**Depth of Inheritance Tree**	**Numerical**
NOC	**Number of Children**	**Numerical**
CBO	**Coupling between Object classes**	**Numerical**
RFC	**Response for a Class**	**Numerical**
LCOM	**Lack of Cohesion in Methods**	**Numerical**
CA	Afferent Couplings	Numerical
CE	Efferent Couplings	Numerical
NPM	Number of Public Methods	Numerical
LCOM3	Normalized version of LCOM	Numerical
LOC	Lines of Code	Numerical
DAM	Data Access Metric	Numerical
MOA	Measure Of Aggregation	Numerical
MFA	Measure of Functional Abstraction	Numerical
CAM	Cohesion Among Methods	Numerical
IC	Inheritance Coupling	Numerical
CBM	Coupling Between Methods	Numerical
AMC	Average Method Complexity	Numerical
MAX_CC	Maximum values of methods in the same class	Numerical

AVG_CC	Mean values of methods in the same class	Numerical

Table 3: Summary of software metrics and antipatterns metrics (shown in bold) for the datasets of each software project.

Abbrev	Feature	Type of Feature
FI.CHANGES	Feature Introduction Changes	Numerical
OSTRAND	Predictor Proposed via OSTRAND et al.	Numerical
SCATTERING	Scattering Metric	Numerical
INTENSITY	Intensity of Code Smells	Numerical
ANA	**Average Number of Antipatterns**	**Numerical**
ACM	**Antipattern Complexity Metric**	**Numerical**
ARL	**Antipattern Recurrence Length**	**Numerical**
ACPD	**Antipattern Cumulative Pairwise Differences**	**Numerical**

3.2 Experimental Methodology

We exploited random forests as a machine learning algorithm in this study [4].

The proposed machine learning approaches are compared against five baselines, as described below.

1) The First Baseline (B1)

This baseline is proposed vi palomba et al. , employing CK metrics, structural metrics (see Table 2), and the other software metrics in Table 3 [29].

2) The Second Baseline (B2)

This baseline is proposed via palomba et al. employing structural metrics (See Table 2) plus intensity metric (see Table 3) [29].

3) The Third Baseline (B3)

This baseline is proposed via palomba et al. utilizing structural metrics (see Table 2) and antipatterns metrics (see Table 3) [29].

4) The Fourth Baseline (B4)

This baseline is proposed via Jureczko et al. utilizing 20 structural metrics in Table 2.

5) The Fifth Baseline (B5)

This baseline employs CK metrics proposed via Chidamber et al. [7].

Table 4 provides an abbreviation summary of top prediction algorithms used in this study. To assess the stability of prediction algorithms, we utilized five-fold cross-validation, where we divided each dataset into five folds, where four folds are used for training a machine learning algorithm while the remaining is used for performing a prediction on the testing. Then, we evaluate the performance using performance results such as accuracy (ACC), precision (PRE), sensitivity (SEN), specificity (SPE), area under curve (AUC), and F1 measure. Such a process is repeated for additional four times, where we report the mean performance results corresponding to MACC,

MPRE, MSEN, MSPE, MAUC, and MF1[18]. Friedman test is exploited in this study to rank the prediction algorithms [5, 6, 8, 13, 16, 34, 39]. Experiments were performed and using R [38].

Table 4: Summary of top prediction algorithms used in our study.

Abbrev	Prediction Algorithm
FML+RF	The first proposed machine learning approach using random forests.
SML+RF	The second proposed machine learning approach using random forests.
B1+RF	The first baseline approach using random forests.
B2+RF	The second baseline approach using random forests.
B3+RF	The third baseline approach using random forests.
B4+RF	The fourth baseline approach using random forests.
B5+RF	The fifth baseline approach using random forests.

3.3 Experimental Results

In this section, we compare the proposed machine learning approaches of software bug prediction against several baselines, reporting 5-fold cross-validation results based on datasets pertaining to two software projects for the task of predicting buggy instances

Predicting Buggy Instances Pertaining to Data of Software Apache Ant Project

Utilization of Data Pertaining to Release 1.3 of Software Apache Ant Project

Table 5 reports prediction results of five prediction algorithms studied in this work. As reported in Table 5, random forests (RF) utilizing the second machine learning approach (SML) achieves higher performance results when compared against baselines, including B1+RF, B2+RF, B3+RF, B4+RF, and B5+RF. Specifically, SML+RF achieves the highest MACC of 0.992, the highest MPRE of 0.950, the highest MSPE of 0.990, the highest MAUC of 0.995, the highest MF1 of 0.974. Although Random forests (RF) utilizing the first baseline approach (B1) performs better than all remaining baselines, random forests utilizing our first machine learning approach FML performs better than B1+RF. These reported results demonstrate the good performance results generated via our approaches.

Table 5: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.3 of software apache ant project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	0.984	0.900	**1.000**	0.981	0.990	0.947
SML+RF	**0.992**	**0.950**	**1.000**	**0.990**	**0.995**	**0.974**
B1+RF	0.944	0.650	**1.000**	0.937	0.968	0.787
B2+RF	0.904	0.450	0.900	0.904	0.902	0.600
B3+RF	0.808	0.200	0.333	0.858	0.596	0.250
B4+RF	0.800	0.200	0.308	0.857	0.582	0.242
B5+RF	0.832	0.250	0.455	0.868	0.661	0.323

Utilization of Data Pertaining to Release 1.4 of Software Apache Ant Project

Table 6 reports performance results of various prediction algorithms when running five-fold cross-validation. It can be seen from Table 6 that a machine learning algorithm (i.e., Random forests) employing our second approach (SML) performs better than all prediction algorithms, including baselines. Particularly, SML+RF generates the highest MACC of 0.988, the highest MPRE of 0.975, the highest MSEN of 0.975, the highest MSPE of 0.992, the highest MAUC of 0.983, and the highest MF1 of 0.975. These high-performance results demonstrate the stability of prediction results generated via SML+RF.

Table 6: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.4 of software apache ant project. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	0.983	0.950	0.974	0.985	0.979	0.962
SML+RF	**0.988**	**0.975**	**0.975**	**0.992**	**0.983**	**0.975**
B1+RF	0.977	0.925	0.973	0.978	0.976	0.948
B2+RF	0.955	0.875	0.921	0.964	0.943	0.897
B3+RF	0.702	0.100	0.190	0.771	0.481	0.131
B4+RF	0.708	0.175	0.269	0.783	0.526	0.212
B5+RF	0.725	0.225	0.333	0.795	0.564	0.269

Utilization of data pertaining to Release 1.5 of Software Apache Ant Project

Table 7 presents performance results of several prediction algorithms when exploiting five-fold cross-validation. The reported results demonstrate the good performance of FML+RF and SML+RF prediction algorithms employing our approaches. In particular, FML+RF and SML+RF generate the highest MACC of 0.989, the highest MPRE of 0.937, the highest MSPE of 0.992, the highest MF1 of 0.952. B2+RF and B1+RF generates a marginal improvement in

terms of MAUC when compared to FML+RF and SML+RF. These performance results demonstrate the good and stable results generated using our approaches.

Table 7: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.5 of software apache ant project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**0.989**	**0.937**	0.967	**0.992**	0.980	**0.952**
SML+RF	**0.989**	**0.937**	0.967	**0.992**	0.980	**0.952**
B1+RF	0.979	0.812	**1.000**	0.977	**0.988**	0.896
B2+RF	0.980	0.813	**1.000**	0.978	**0.989**	0.897
B3+RF	0.867	0.094	0.231	0.896	0.564	0.133
B4+RF	0.891	0.281	0.500	0.916	0.708	0.360
B5+RF	0.884	0.219	0.438	0.910	0.674	0.292

Utilization of data pertaining to Release 1.6 of Software Apache Ant Project

Table 8 shows performance results generated via several prediction algorithms when employing five-fold cross-validation. It can be shown from Table 8 that FML+RF and SML+RF generate perfect results as reported via various performance measures including MACC, MPRE, MSEN, MSPE, MAUC, and MF1. Compared to several baselines, these results show the superiority of random forests when employing our approaches, FML and SML.

Table 8: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.6 of software apache ant project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
SML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
B1+RF	0.994	0.989	0.989	0.996	0.992	0.989
B2+RF	0.997	0.989	**1.000**	0.996	0.998	0.995
B3+RF	0.818	0.587	0.675	0.860	0.767	0.628
B4+RF	0.815	0.598	0.663	0.862	0.762	0.629
B5+RF	0.795	0.500	0.639	0.835	0.737	0.561

Utilization of data pertaining to Release 1.7 of Software Apache Ant Project

Table 9 reports performance results of prediction algorithms when utilizing five-fold cross-validation. The reported performance results on Table 9 show that FML+RF and SML+RF outperform all prediction algorithms including baselines. Specifically, FML+RF and SML+RF generate the highest MACC of 0.998, the highest MPRE of 1, the highest MSEN of 0.994, the highest MSPE of 1, the highest MAUC of 0.997, and the highest MF1 of 0.996. These high generated performance results reflect the stability of random forests when incorporated with our proposed approaches.

Table 9: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.7 of software apache ant project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**0.998**	**1.000**	**0.994**	**1.000**	**0.997**	**0.996**
SML+RF	**0.998**	**1.000**	**0.994**	**1.000**	**0.997**	**0.996**
B1+RF	0.997	0.993	0.993	0.998	0.996	0.993
B2+RF	0.997	0.994	**0.994**	0.998	0.996	0.994
B3+RF	0.823	0.434	0.655	0.852	0.753	0.522
B4+RF	0.821	0.458	0.639	0.856	0.747	0.533
B5+RF	0.796	0.392	0.560	0.839	0.700	0.461

Predicting Buggy Instances Pertaining to Data of Software Apache Camel Project

Utilization of Data Pertaining to Release 1.0 of Software Apache Camel Project

Table 10: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.0 of software apache camel project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**0.994**	**0.846**	**1.000**	**0.993**	**0.996**	**0.916**
SML+RF	**0.994**	**0.846**	**1.000**	**0.993**	**0.996**	**0.916**
B1+RF	0.961	0.076	0.500	0.964	0.732	0.133
B2+RF	0.965	0.154	0.667	0.967	0.817	0.250
B3+RF	0.956	0.000	0.000	0.961	0.481	--
B4+RF	0.956	0.000	0.000	0.961	0.481	--
B5+RF	0.956	0.000	0.000	0.961	0.481	--

Table 10 reports performance results generated by prediction algorithms employing our approaches and baselines. Table 10 shows that prediction algorithm FML+RF and SML+RF employing our approaches generate the highest MACC of 0.994, the highest MPRE of 0.846, the highest MSEN of 1, the highest MSPE of 0.993, the highest MAUC of 0.996, the highest MF1 of 0.916. These generated performance results demonstrate the good performance of random forests when utilizing our approaches.

Utilization of Data Pertaining to Release 1.2 of Software Apache Camel Project

Table 11 reports performance result generated via several prediction algorithms. It can be seen from Table 11 that prediction algorithm SML+RF utilizing our second machine learning approach generates the highest performance results. These generated performance results demonstrate the effectiveness of SML+RF.

Table 11: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.2 of software apache camel project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	0.995	0.986	**1.000**	0.992	0.996	0.993
SML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
B1+RF	0.993	0.981	**1.000**	0.989	0.994	0.990
B2+RF	0.997	0.991	**1.000**	0.995	0.997	0.995
B3+RF	0.671	0.296	0.571	0.694	0.632	0.390
B4+RF	0.674	0.324	0.574	0.700	0.637	0.414
B5+RF	0.623	0.292	0.453	0.674	0.564	0.355

Utilization of Data Pertaining to Release 1.4 of Software Apache Camel Project

In Table 12, the performance measures of several prediction algorithms are reported using five-fold cross-validation. Table 12 shows that prediction algorithms FML+RF and SML+RF exploiting our approaches outperform all prediction algorithms and generate perfect performance results. These results demonstrate the stable performance of our approaches.

Table 12: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.4 of software apache camel project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
SML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
B1+RF	0.997	0.986	**1.000**	0.997	0.998	0.993
B2+RF	0.999	0.993	**1.000**	0.999	0.999	0.997
B3+RF	0.844	0.152	0.629	0.853	0.741	0.244
B4+RF	0.846	0.179	0.634	0.857	0.745	0.280
B5+RF	0.827	0.117	0.425	0.846	0.636	0.184

Utilization of Data Pertaining to Release 1.6 of Software Apache Camel Project

Table 13 reports performance results of prediction algorithms when utilizing five-fold cross-validation. It can be seen from Table 13 that when performance measures including MACC, MPRE, MSEN, MSPE, MAUC, MF1 are used, the prediction algorithms employing FML+RF and SML+RF generate perfect performance results, which reflect the effective mechanism employed via our approaches compared to the baselines.

Table 13: Performance results of studied prediction algorithms utilizing five-fold cross-validation on release 1.6 of software apache camel project data. The highest performance results are shown in bold. MACC denotes the mean accuracy. MPRE denotes the mean precision. MSEN denotes mean sensitivity. MSPE denotes mean specificity. MAUC denotes mean area under curve. MF1 denotes mean F1 measures.

	MACC	MPRE	MSEN	MSPE	MAUC	MF1
FML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
SML+RF	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
B1+RF	0.996	0.984	**1.000**	0.996	0.998	0.991
B2+RF	0.999	0.995	**1.000**	0.999	0.999	0.997
B3+RF	0.800	0.101	0.442	0.817	0.629	0.165
B4+RF	0.811	0.213	0.541	0.834	0.687	0.305
B5+RF	0.797	0.154	0.439	0.823	0.631	0.228

4 Discussion

The proposed machine learning approaches aim to select important features to guide the prediction process. The second machine learning (SML) approach is considered as an extension of the first machine learning (FML) approach. The only difference is that SML utilizes K-means in the data preparation step. Our experimental results on several datasets pertaining to Software Apache Ant and Camel projects demonstrate the superior performance results of our approaches in terms of accurate prediction results when compared to the prediction algorithms employing baselines.

According to the experimental results, random forests coupled with the second machine learning approach, RF+SML, performed better than the baseline prediction algorithms. This demonstrates that our approaches select a subset of features which play an important role for improving the prediction performance. Moreover, these features could be used as markers to guide software developers in detecting buggy instances.

In the second and third baseline approaches (i.e., B2 and B3) proposed by Palomba et al, they removed 4 metrics (RFC, CA, LCOM, and MAX_CC) from the 20 structured metrics. The reason is that these 4 metrics are highly correlated with WMC, CBO, LCOM3, and AVG_CC. Hence, B2 and B3 utilize 16 metrics out of the 20 [29].
It is worth mentioning that we assessed the performance of several variants of neural networks as well as support vector machines [33, 14]. However, they did not exhibit good performance results; therefore, their results are not included in this paper.

5 Conclusion and Future Work

Two machine learning approaches are proposed to improve software bug prediction. The first approach (FML) aims to identify important software metrics via utilizing an ensemble of decision trees, to extract the important metrics. Then, a machine learning algorithm is trained on a subset of metrics to generate a model and perform predictions

on unseen examples. The second approach is an extension of the FML approach, in which K-means is utilized to partition training data into two clusters. Then, each training example located in a given cluster is provided to an ensemble of decision trees, to extract important software metrics. Next, a machine learning algorithm takes as input the training set, which is the union of the subsets of features obtained from each ensemble, to generate a model. Finally, the model is used to perform predictions on test examples. Compared to existing prediction algorithms employing several baselines, experimental results on several datasets pertaining to software projects demonstrate the superiority of our approaches in terms of high-performance results.

In future work, we aim to (1) utilize the proposed approaches for identifying important features in different domains including biology and medicine; (2) incorporate the proposed approaches in the transfer learning settings to identify important features in related tasks.

References

[1] V. R. BASILI, L. C. BRIAND and W. L. MELO, *A validation of object-oriented design metrics as quality indicators*, IEEE Transactions on software engineering, 22 (1996), pp. 751-761.

[2] R. M. BELL, T. J. OSTRAND and E. J. WEYUKER, *Does measuring code change improve fault prediction?*, *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, ACM, Banff, Alberta, Canada, 2011, pp. 1-8.

[3] R. M. BELL, T. J. OSTRAND and E. J. WEYUKER, *The limited impact of individual developer data on software defect prediction*, Empirical Software Engineering, 18 (2013), pp. 478-505.

[4] L. BREIMAN, *Random forests*, Machine learning, 45 (2001), pp. 5-32.

[5] D. BRZEZINSKI and J. STEFANOWSKI, *Reacting to different types of concept drift: The accuracy updated ensemble algorithm*, IEEE Transactions on Neural Networks and Learning Systems, 25 (2014), pp. 81-94.

[6] B. CALVO and G. SANTAFE RODRIGO, *scmamp: Statistical comparison of multiple algorithms in multiple problems*, The R Journal, Vol. 8/1, Aug. 2016 (2016).

[7] S. R. CHIDAMBER and C. F. KEMERER, *A metrics suite for object oriented design*, IEEE Transactions on software engineering, 20 (1994), pp. 476-493.

[8] J. DEMŠAR, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine learning research, 7 (2006), pp. 1-30.

[9] D. DI NUCCI, F. PALOMBA, G. DE ROSA, G. BAVOTA, R. OLIVETO and A. DE LUCIA, *A developer centered bug prediction model*, IEEE Transactions on Software Engineering, 44 (2018), pp. 5-24.

[10] K. EL EMAM, W. MELO and J. C. MACHADO, *The prediction of faulty classes using object-oriented design metrics*, Journal of Systems and Software, 56 (2001), pp. 63-75.

[11] J. EYOLFSON, L. TAN and P. LAM, *Do time of day and developer experience affect commit bugginess?*, *Proceedings of the 8th Working Conference on Mining Software Repositories*, ACM, 2011, pp. 153-162.

[12] S. FAN, X. LI and J. L. ZHAO, *Collaboration process pattern approach to improving teamwork performance: A data mining-based methodology*, INFORMS Journal on Computing, 29 (2017), pp. 438-456.

[13] S. GARCíA, A. FERNáNDEZ, J. LUENGO and F. HERRERA, *Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power*, Information Sciences, 180 (2010), pp. 2044-2064.

[14] I. GOODFELLOW, Y. BENGIO and A. COURVILLE, *Deep learning*, MIT press, 2016.

[15] T. L. GRAVES, A. F. KARR, J. S. MARRON and H. SIY, *Predicting fault incidence using software change history*, IEEE Transactions on Software Engineering, 26 (2000), pp. 653-661.

[16] D. C. HOWELL, *Fundamental statistics for the behavioral sciences*, Nelson Education, 2016.

[17] A. K. JAIN, M. N. MURTY and P. J. FLYNN, *Data clustering: a review*, ACM computing surveys (CSUR), 31 (1999), pp. 264-323.

[18] N. JAPKOWICZ and M. SHAH, *Evaluating Learning Algorithms: A Classification Perspective*, Cambridge University Press, 2011.

[19] M. JURECZKO and L. MADEYSKI, *Towards identifying software project clusters with regard to defect prediction*, *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, ACM, 2010, pp. 9.

[20] C. LEWIS, Z. LIN, C. SADOWSKI, X. ZHU, R. OU and E. J. WHITEHEAD JR, *Does bug prediction support human developers? findings from a google case study*, *Proceedings of the 2013 international conference on Software Engineering*, IEEE Press, 2013, pp. 372-381.

[21] Q. MI, J. KEUNG, Y. HUO and S. MENSAH, *Not all bug reopens are negative: A case study on eclipse bug reports*, Information and Software Technology, 99 (2018), pp. 93-97.

[22] R. MOSER, W. PEDRYCZ and G. SUCCI, *Analysis of the reliability of a subset of change metrics for defect prediction*, *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ACM, Kaiserslautern, Germany, 2008, pp. 309-311.

[23] R. MOSER, W. PEDRYCZ and G. SUCCI, *A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction*, *Proceedings of the 30th international conference on Software engineering*, ACM, Leipzig, Germany, 2008, pp. 181-190.

[24] N. NAGAPPAN, T. BALL and A. ZELLER, *Mining metrics to predict component failures*, *Proceedings of the 28th international conference on Software engineering*, ACM, Shanghai, China, 2006, pp. 452-461.

[25] N. NAGAPPAN, T. BALL and A. ZELLER, *Mining metrics to predict component failures*, *Proceedings of the 28th international conference on Software engineering*, ACM, 2006, pp. 452-461.

[26] J. NAM, W. FU, S. KIM, T. MENZIES and L. TAN, *Heterogeneous defect prediction*, IEEE Transactions on Software Engineering, 44 (2018), pp. 874-896.

[27] D. D. NUCCI, F. PALOMBA, D. A. TAMBURRI, A. SEREBRENIK and A. D. LUCIA, *Detecting code smells using machine learning techniques: Are we there yet?*, *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 612-621.

[28] T. J. OSTRAND, E. J. WEYUKER and R. M. BELL, *Programmer-based fault prediction*, *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, ACM, 2010, pp. 19.

[29] F. PALOMBA, M. ZANONI, F. A. FONTANA, A. DE LUCIA and R. OLIVETO, *Toward a smell-aware bug prediction model*, IEEE Transactions on Software Engineering (2017).

[30] S. K. PANDEY, R. B. MISHRA and A. K. TRIPHATHI, *Software Bug Prediction Prototype Using Bayesian Network Classifier: A Comprehensive Model*, Procedia computer science, 132 (2018), pp. 1412-1421.

[31] D. POSNETT, R. D, #039, SOUZA, P. DEVANBU and V. FILKOV, *Dual ecological measures of focus in software development*, *Proceedings of the 2013 International Conference on Software Engineering*, IEEE Press, San Francisco, CA, USA, 2013, pp. 452-461.

[32] S. RIAZ, A. ARSHAD and L. JIAO, *Rough Noise-Filtered Easy Ensemble for Software Fault Prediction*, IEEE Access, 6 (2018), pp. 46886-46899.

[33] B. SCHOLKOPF and A. J. SMOLA, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT press, 2001.

[34] L. SHANG, Z. LU and H. LI, *Neural responding machine for short-text conversation*, arXiv preprint arXiv:1503.02364 (2015).

[35] S. SHIVAJI, E. J. WHITEHEAD, R. AKELLA and S. KIM, *Reducing features to improve code change-based bug prediction*, IEEE Transactions on Software Engineering, 39 (2013), pp. 552-569.

[36] R. SUBRAMANYAM and M. S. KRISHNAN, *Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects*, IEEE Transactions on software engineering, 29 (2003), pp. 297-310.

[37] S. E. S. TABA, F. KHOMH, Y. ZOU, A. E. HASSAN and M. NAGAPPAN, *Predicting bugs using antipatterns*, *2013 IEEE International Conference on Software Maintenance*, IEEE, 2013, pp. 270-279.

[38] R. C. TEAM, *R: A language and environment for statistical computing*, (2013).

[39] T. TURKI and Y. TAGUCHI, *Machine Learning Algorithms for Predicting Drugs–Tissues Relationships*, Expert Systems with Applications (2019).

[40] S. ZAMAN, B. ADAMS and A. E. HASSAN, *Security versus performance bugs: a case study on firefox*, *Proceedings of the 8th working conference on mining software repositories*, ACM, 2011, pp. 93-102.

[41] Y. ZHOU, B. XU and H. LEUNG, *On the ability of complexity metrics to predict fault-prone classes in object-oriented systems*, Journal of Systems and Software, 83 (2010), pp. 660-674.

[42] X. ZHU, B. NIU, E. J. WHITEHEAD JR and Z. SUN, *An empirical study of software change classification with imbalance data-handling methods*, Software: Practice and Experience, 48 (2018), pp. 1968-1999.

How is Your Team Spirit? Cluster Over-Time Stability Evaluation

Martha Tatusch*[0000−0001−6302−6070], Gerhard Klassen*[0000−0002−1458−6546], Marcus Bravidor[0000−0003−1504−9889], and Stefan Conrad[0000−0003−2788−3854]

Heinrich Heine University, Universitätsstr. 1, 40225 Düsseldorf, Germany
{tatusch,klassen,bravidor,stefan.conrad}@hhu.de

Abstract. Clustering of time series data is a major part of data mining. In this paper, we consider multiple multivariate time series and the clustering of their data points per timestamp. One of the major problems of this approach is that the temporal connection of clusterings at different times can neither be guaranteed nor tracked. For this reason we present CLOSE (**Cl**uster **O**ver-Time **S**tability **E**valuation): an internal evaluation measure for clusterings of temporal data. Our method evaluates not only the quality but also the over-time stability of the clusters. Time series with an equal cluster neighborhood over time are considered to be stable while those which change their neighbors often are considered as unstable. We applied our model to different data and present the results in this paper.

Keywords: Time Series Analysis · Clustering · Evaluation

1 Introduction

Information extraction from time series (TS) is well researched. There are many different approaches which all tackle specific problems. Often clustering the data has an important fraction in the concept of choice. While some of those methods divide the time series in parts, so called subsequences [2], others consider the whole time series at once [19], yet others extract feature sets [10, 26]. Although these approaches seem to solve a lot of problems and enable the discovery of knowledge, new problems like the choice of parameters arise. This parameter choice often ends up with many apparently good solutions and lacks an evaluation function which distinguishes the quality of clusterings properly. This problem grows with the amount of dimensions and requires an automatic rating of the available solutions.

In this paper we consider multiple multivariate time series with same length and equivalent time steps. We detect clusters for each point in time (called over-time clustering) with different parameters and identify the best overall clustering without knowing the ground truth. Therefore, we present an internal evaluation measure for temporal clusterings which can be used to rate and compare different clustering results of time series data. Our method, which is named CLOSE

* Both authors contributed equally to this research.

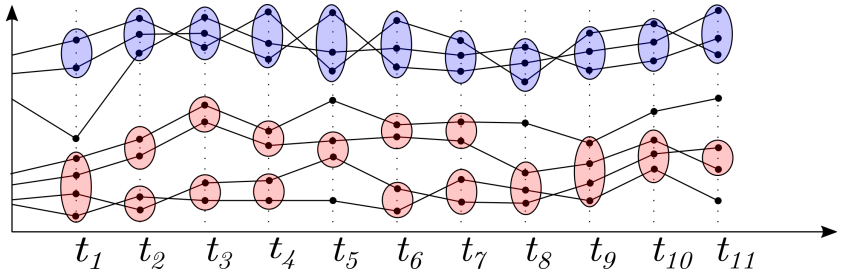


Fig. 1: Example of a time series over-time clustering [25]. The red clusters are less stable over time than the blue ones.

(**Cl**uster **O**ver-Time **S**tability **E**valuation), not only evaluates the quality of the individual clusterings per time point, but also the over-time stability. The temporal aspect is thus included in the evaluation. For the first time, this makes it possible to rate a clustering of time series data, in which the data points are clustered per timestamp, regarding the temporal linkage of clusters. Furthermore, the presented method is able to handle missing data points without adaptation. An example of an over-time clustering is illustrated in Figure 1. For a simple visualization, univariate time series are shown. Compared to whole time series clustering, this technique has a major advantage: similar partial sequences of undefined length can be found.

This approach is not only novel in the sense that it considers quality and over-time stability at the same time, but also because over-time stability differs from the stability usually represented in literature. It serves a different purpose and is based on transitions between clusters over time, which will be explained in more detail later in this work. The procedure for example may be useful when tracking topics in online forums. By clustering per point in time, the development of relationships between different terms can be investigated. When examining financial data, the procedure can lead to a gain in information as well. Assuming that the courses of different companies' financial data can be divided into groups – e.g. successful and less successful companies – clustering might be helpful to detect anomalies or even fraud. Since it cannot be guaranteed that all fraud cases are known – some may remain uncovered – this problem cannot be solved with fully supervised learning. The identification of meaningful groups would be a fundamental step. In General, the evaluation of temporal clusterings enables the identification of suitable hyper-parameters for different algorithms as a basis for further analysis such as outlier detection.

In the further course we will first discuss similar work (Section 2). We then define the considered problem (Section 3) and present our solution (Section 4). Finally, we give an overview of our experiments and their results (Section 5), discuss the method (Section 6) and draw a conclusion (Section 7).

2 Related Work

To the best of our knowledge, there does not exist any approach similar to ours, since clustering evaluation metrics usually do not contain a temporal component. For this reason, we refer on the one hand to related work with regard to time series clustering and on the other hand to time-independent evaluation of clusterings.

2.1 Time Series Clustering

In the field of time series analysis, there are different techniques for clustering time series data. When considering multiple time series, one approach is the clustering of the entire sequences [7, 19]. For our context, this procedure is not well suited as potential correlations between subsequences of different time series are not revealed. Additionally, the exact course of the time series is not relevant, but rather the trend they show. The transformation of entire sequences to feature vectors, which then are clustered [10], blurs the exact course and is a popular method. Still, the problem of not recognizing interrelated subsequences persists.

However, there is also the approach of clustering subsequences of a time series [2, 12]. Usually, this is done to find motifs in time series and therefore only a single time series is considered. In [14], Keogh et al. state that the clustering of subsequences of a single time series is meaningless, though. However, this statement is controversial, as Chen [4] argues that it is possible to obtain meaningful results if the correct distance measure is used. For this purpose, various distance measures have been introduced [23, 24].

There is also the approach of clustering partial sequences of multiple time series. Outliers may influence the results, though, and there is a need of finding a meaningful length of the subsequences, since the examination of subsequences of all lengths is usually very time-consuming. Our approach can provide more insights as subsequences of any length can selectively be investigated. However, under the assumption that the entire time course from the beginning is relevant, CLOSE only considers subsequences starting at the first point in time.

Methods for the clustering of streaming data [9, 18] are not comparable to our method, as they consider only one time series at a time and deal with other problems such as high memory requirements and time complexity.

2.2 Internal Evaluation Measures

There are many different evaluation measures for evaluating clusters and clusterings. Thereby, a distinction between *external* and *internal* measures ought to be made. In the case of the external evaluation, the ground truth is already known so that the results can be compared with expectations. In the internal evaluation, no information about the actual classes is known, so that the clusters are evaluated primarily on the basis of characteristics such as compactness or separation.

One metric that evaluates the compactness of clusters is the *Sum of Squared Errors*. It calculates the overall distance between the members and the *centroid* of a cluster. The centroid is usually the mean of all cluster members. The closer the objects of a cluster lie together, the smaller the error, the greater the compactness. However, this measure does not take into account the separation of different clusters.

The *Silhouette Coefficient* [22] evaluates the compactness as well as the separation of different clusters. This is achieved by using both the average distance of an object to members of its cluster and the average distance to members of the nearest cluster. These two properties are also addressed in the *Davies-Bouldin Index* [5] and the *Dunn Index* [6].

All these metrics cannot be directly compared to our method since they lack a temporal aspect. However, as we will show in the following, they can be applied in CLOSE.

2.3 Stability Evaluation

For the stability measurement of a clustering algorithm there are already several methods. The *Rand Index* [20], which is usually intended for the external evaluation of a clustering, can e.g. be used for this purpose. This evaluation measure rates the agreement of a clustering ζ_p with the expected result ζ_t (ground truth). Therefore it examines all object pairs that are located in the same cluster in ζ_p as well as ζ_t and all pairs that belong to different clusters in both clusterings. The number of corresponding object pairs is then set in relation to the number of all possible object pairs. Considering n objects, the number of all possible pairs is $\binom{n}{2}$.

Measuring the stability of a clustering algorithm is for instance made in order to find the optimal k for KMeans [17] or to determine the dependence of a clustering on its initialization. When considering m clusterings ζ_i ($1 \leq i \leq m$) with the same parameter k and random initialization, the Rand Index is calculated for every unordered pair of clusterings ζ_i, ζ_j with $i \neq j$ by assuming ζ_i is the ground truth without loss of generality. The stability is expressed by the average Rand Index across all pairs. Such stability measures, however, pursue a different objective and clearly do not take a temporal linkage into consideration [16].

An obvious idea would be to measure over-time stability by comparing clustering pairs of successive points in time. However, this approach is inflexible and would strongly weight variation between two points in time, although the clustering might deviate only at one point in time and otherwise remain quite stable. An ongoing change, on the other hand, would be punished only very slightly, since the variation between clusterings of two adjacent timestamps would be small, but in regard to the entire period the change would be very large. Furthermore a separation or merge of clusters would have a strong negative impact on the index. Even when comparing all possible clustering pairs of different time points these problems would persist. Our method handles such cases in a slightly different way.

In addition, the Rand Index exclusively evaluates the (over-time) stability of a clustering. But as stated in [3, 15], stability alone does not imply a *good* clustering. If this is not the case with constant data points, then certainly it is not the case with data points that change over time. CLOSE combines the evaluation of the over-time stability and the quality of a clustering to give an overall statement about an over-time clustering. However, changing values of the data objects is another problem that has to be faced when looking at time series data.

The identification of so called *Moving Clusters* [13] seems to be a closely related topic, but addresses a slightly different problem. In contrast to the evaluation of an over-time clustering, this field of research deals with the detection of clusters that remain mostly the same in regard to their members. In [13] an intuitive approach using the Jaccard Index is presented for the problem. If the Jaccard Index of two clusters of different timestamps is greater than θ, these clusters are identified as the *same cluster* for different timestamps. Apart from the fact that the clustering is not evaluated here, there is another difference to our approach: it is assumed that a cluster remains approximately the same size over time. In real data, however, this is not necessarily the case. This may apply to some tasks, such as herd tracking, which is examined in the paper, but in most cases this requirement is not satisfied.

3 Fundamentals

Since there are various approaches and definitions concerning TS analysis, we next clarify our understanding of some basic concepts regarding our approach.

Definition 1 (Time Series). *A time series $T = o_{t_1}, ..., o_{t_n}$ is an ordered set of n real valued data points of arbitrary dimension. The data points are chronologically ordered by their time of recording, with t_1 and t_n indicating the first and last timestamp, respectively.*

Definition 2 (Data Set). *A data set $D = T_1, ..., T_m$ is a set of m time series of same length n and equivalent points in time.*

The vectors of all time series are denoted as the set $O = \{o_{t_1,1}, ..., o_{t_n,m}\}$. With the second index indicating the time series the data point originates from. We write O_{t_i} for all data points at a certain point in time.

Definition 3 (Cluster). *A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $j \in \{1, ..., p\}$ being an unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm. This means that all clusters have distinct labels regardless of time.*

Definition 4 (Cluster Member). *A data point $o_{t_i,l}$ at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.*

Definition 5 (Noise). *A data point $o_{t_i,l}$ at time t_i is considered as noise, if it is not assigned to any cluster. A data point that belongs to noise is also called an outlier.*

Definition 6 (Clustering). *A clustering is the overall result of a clustering algorithm for all timestamps. In concrete it is the set $\zeta = \{C_{t_1,1}, ..., C_{t_n,p}\} \cup Noise$.*

4 Method

A major disadvantage of creating clusters for every timestamp is an evident missing temporal link. In our approach we assume that different clusterings deliver different cluster connectedness and that this bond can be measured. In order to measure the temporal linking we make use of a stability function. Given a clustering ζ, we first analyze the behavior of every subsequence of a time series $T = o_{t_1}, ...o_{t_k}$, with $t_k \leq t_n$, starting at the first timestamp. This is done, because time series which separate from their clusters' members often, indicate a low temporal linkage. One could say we evaluate the *team spirit* of the individual time series. Further, we rate every cluster with a stability function, which depends on the subsequence analysis and the number of clusters merged into this cluster. Finally, we assign a score to the clustering, depending on the over-time stability of every cluster.

Let $C_{t_i,a}$ and $C_{t_j,b}$ be two clusters, with $t_i, t_j \in \{t_1, ...t_n\}$. In order to measure the stability of a time series we first introduce the *temporal cluster intersection*

$$\cap_t\{C_{t_i,a}, C_{t_j,b}\} = \{T_l \mid o_{t_i,l} \in C_{t_i,a} \wedge o_{t_j,l} \in C_{t_j,b}\}, \tag{1}$$

with $l \in \{1, ..., m\}$. The temporal cluster intersection returns a set of time series, which contain data points grouped together in t_i as well as in t_j. Now the behavior of a subsequence from one cluster $C_{t_i,a}$ in t_i to another $C_{t_j,b}$ in t_j can be expressed by the proportion of members of $C_{t_i,a}$ remaining together in t_j

$$p(C_{t_i,a}, C_{t_j,b}) = \frac{|C_{t_i,a} \cap_t C_{t_j,b}|}{|C_{t_i,a}|}, \tag{2}$$

with $t_i < t_j$. In the example in Figure 2 the proportion for $C_{t_i,l}$ and $C_{t_j,v}$ would be

$$p(C_{t_i,l}, C_{t_j,v}) = \frac{|\{a,b\}|}{|\{a,b\}|} = \frac{2}{2} = 1.0.$$

With the help of the proportion of clusters we now can rate all data points of a sequence with a *subsequence score*. It is defined as

$$subseq\_score(o_{t_k,l}) = \frac{1}{k_a} \cdot \sum_{i=1}^{k-1} p(cid(o_{t_i,l}), cid(o_{t_k,l})), \tag{3}$$

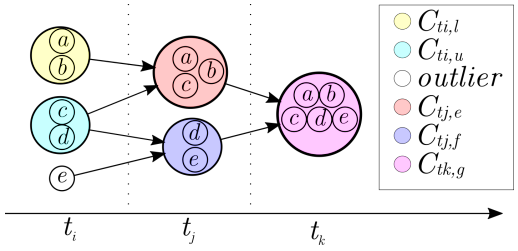


Fig. 2: Example for transitions of TS $a, .., e$ between clusters over time [25].

with $l \in \{1, ..., m\}$, $k_a \in [1, k-1]$ being the number of timestamps where the data point exists and is assigned to a cluster (therefore is **not** recognized as noise), and *cid*, the cluster-identity function

$$cid(o_{t_i,j}) = \begin{cases} \emptyset & \text{if the data point is not assigned to a cluster} \\ C_{t_i,l} & \text{else} \end{cases} \tag{4}$$

returning the cluster which the data point has been assigned to in t_i. Thus, in this equation, all time points in which an object is an outlier, are ignored. The *subsequence score* takes into account how many objects from the previous clusters have migrated together with the currently viewed object.

Regarding the example of Figure 2, the score of time series a in time point t_k would be:

$$subseq\_score(o_{t_k,a}) = \frac{1}{2} \cdot (1.0 + 1.0) = 1.0.$$

This value reflects the highest stability. The time series d, on the other hand, gets a lower value of $subseq\_score(o_{t_k,d}) = 0.75$ as it once changes the cluster without its cluster members.
The rating of clusters depends on two factors. The first factor is the number of merged clusters

$$m(C_{t_k,i}) = |\{C_{t_l,j} \mid t_l < t_k \wedge \exists a : o_{t_l,a} \in C_{t_l,j} \wedge o_{t_k,a} \in C_{t_k,i}\}|, \tag{5}$$

which describes the amount of different clusters of previous timestamps, that merged into the regarded cluster. The second factor is the sum of all *subsequence scores* of the data points within the regarded cluster. So the *over-time stability* of a cluster is defined as

$$ot\_stability(C_{t_k,i}) = \frac{\frac{1}{|C_{t_k,i}|} \cdot \sum_{o_{t_k,l} \in C_{t_k,i}} subseq\_score(o_{t_k,l})}{\frac{1}{k-1} \cdot m(C_{t_k,i})} \tag{6}$$

for $k > 1$. Note that the entire preceding time frame is considered. For the first timestamp we consider clusters to be stable and set $ot\_stability(C_{t_1,i}) = 1.0$. It is important to mention, that the number of merged clusters does not take outliers into account.

Regarding the example of Figure 2, the stability of the cluster $C_{t_k,g}$ would be:

$$ot\_stability(C_{t_k,g}) = \frac{\frac{1}{5} \cdot (1.0 + 1.0 + 0.75 + 0.75 + 1.0)}{\frac{1}{2} \cdot 4} = 0.45.$$

This low score can be explained by the fact that the cluster under consideration contains only a few data points, two of which already have an independent course of their clusters' members.
Finally we can rate the over-time stability of a clustering ζ:

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \Big(1 - \Big(\frac{k}{N_C}\Big)^2\Big) \cdot \Big(\sum_{C \in \zeta} ot\_stability(C) \cdot (1 - quality(C))\Big), \quad (7)$$

with N_C being the number of clusters of the whole clustering, k being the number of considered timestamps and *quality* being an arbitrary cluster rating function. We suggest the mean squared error (MSE) but density ratings like the local outlier factor (LOF) can also be used. Be aware using a function in the interval of $[0, 1]$ in order to get appropriate results. If greater values indicate a higher quality, $(1-quality(C))$ may e.g. be replaced by $(1-quality(C)^{-1})$ or $quality(C)$ depending on the quality measure.

When using normalized data with feature values in $[0, 1]$, and a measure function in $[0, 1]$, CLOSE as well returns a score between 0 and 1, with 1 indicating a good over-time clustering, as long as there is at least one cluster per timestamp.

The pre-factors result on the one hand from averaging by the number of clusters and on the other hand from the factor $1 - (\frac{k}{N_C})^2$. This is intended to counteract one large cluster, since such a clustering automatically receives a very high rate of over-time stability. The more clusters exist per time, the larger the factor. However, to prevent the creation of too many clusters, the influence of the fraction is diminished by squaring it.

Remark 1 (Time Point Comparison). In contrast to comparing pairs of consecutive points in time, CLOSE contains temporal information that is robust against outliers. By comparing clusterings of all preceding time points with the last timestamp of the considered subsequence, short-term changes to other clusters are weighted more lightly. In addition, long-term changes that develop slowly over time are punished more severely. Since the influence of the *over-time stability* is weighted with the *quality* of the cluster, the formula cannot be transformed to simply iterate over all cluster pairs.

Remark 2 (Handling Outliers). Our calculations are suitable for both cleaned and noisy data. Since outliers are neither considered in the subsequence score nor in the cluster stability, they have no influence at this point. However, they do have an indirect influence on the calculation of the clustering score. The prefactor favors a large number of clusters. Depending on the quality of the clusters, it may be more advantageous for the algorithm to assign data points to smaller clusters than to interpret them as noise and recognize only a few large clusters.

In view of the fact that over-time clustering might be used for outlier detection, this treatment of outliers is reasoned. In this case, the algorithm should not

be forced to assign every data object to a cluster. Nevertheless, the treatment of outliers may be extended in future work. One way to penalize noise would be, to replace k_a in the subsequence score with k. This would cause, that outliers would get the worst score of 0, as the timestamps would not be skipped.

Remark 3 (Merge & Split of Clusters). Considering the *subsequence score*, a merge of clusters has no negative impact on the score. On the contrary: if two clusters fuse entirely, the score is actually very good, since all objects move with all their cluster members. This circumstance is intended, as the focus is primarily on the cohesion of time series. As long as a group of time series remains together, it is not negative if more are joining.

If a split happens, however, the *subsequence score* decreases. This is also wanted, as a split indicates that time series that have formed a group at one point in time no longer hold together. This fact contradicts the desired cohesion and will be penalized in any case. If smaller clusters have previously been merged and then separated again in the same way as before, this has no great influence on the score over time, though.

Remark 4 (Additional Remark). A small sample size not only influences the stability when considering constant data points [3], but also leads to a high sensitivity to transitions between clusters when examining the over-time stability. The more data points are considered, the easier it is to give a meaningful statement about the (over-time) stability.

5 Experiments

To the best of our knowledge there are no comparable measures presented in literature. This is why we decided to make experiments to demonstrate the results of our measure. We show the transferability of our method to reality by performing two experiments on real data. Additionally, we present the results on two artificially generated data sets, that satisfy the necessary assumptions for the meaningful use of CLOSE, to show the impact of the over-time stability. For all experiments MSE was chosen as the cluster quality measure.

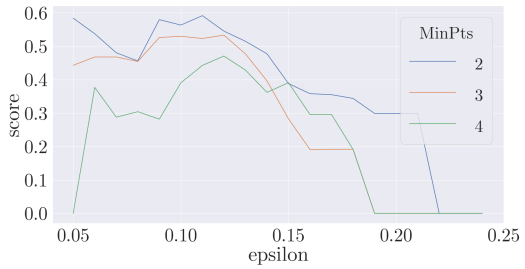


Fig. 3: Achieved CLOSE scores for $minPts \in [2, 4]$ depending on ϵ on the EIKON Financial data set.

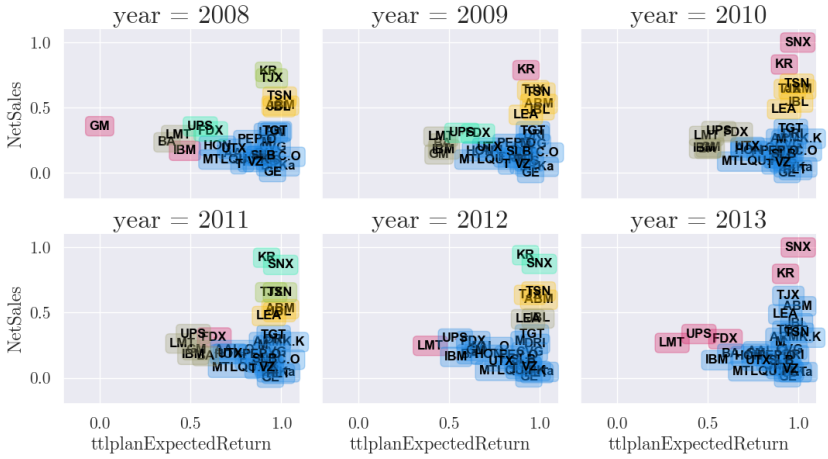


Fig. 4: Detected clusters by DBSCAN with $minPts = 2$ and $\epsilon = 0.11$ on the *EIKON Financial* data set. Red data points represent outliers.

5.1 EIKON Financial Data Set

The first data set is extracted from *EIKON* [21] which is a commercial set of software products released by Refinitiv (formerly Thomson Reuters Financial & Risk). It includes a database with financial information of thousands of companies. For the ease of visibility we chose two random features of fifty random companies. The features we chose are the net sales and the total plan expected return, which are figures taken from the balance sheet of the companies. Thomson Reuters named the according fields *TR-NetSales* and *TR-TtlPlanExpectedReturn*, respectively. The first feature represents the sales receipts for products and services without cash discounts, trade discounts, excise tax, sales returns and allowance. The second feature represents the total amount of expected return on all of a company's pension and post-retirement plans. We normalized the data through dividing the features by the total assets. This is a common approach in economics. The coefficient of correlation of these two features regarding our subset is 0.210. One time series represents the described features of one company over time.

In order to evaluate the CLOSE score on this data set we used the clustering algorithm DBSCAN [8] and applied a grid search with three different $minPts$ (2,3,4) in the epsilon interval $[0.05, 0.25]$. In Figure 3 it can be seen, that $minPts = 2$ reached the maximal CLOSE score of 0.59 at $\epsilon = 0.11$. Clusterings with $minPts = 3$ and $minPts = 4$ reached lower scores at higher epsilons. This is an expected behavior, since a higher $minPts$ would require a higher ϵ in order to create a cluster in this data set. A higher ϵ leads to a higher mse, which has a negative effect on the CLOSE score.

The resulting clustering is illustrated in Figure 4 and shows a very stable clustering. Especially notable is the subsequence from 2008 to 2012, which shows only minor variations.

5.2 GlobalEconomy Data Set

The second dataset is obtained from www.theglobaleconomy.com [1], which is a website that provides economic data for different countries. For this experiment we randomly selected two features, namely the "Unemployment Rate" and the "Public spending on education, percent of GDP". In order to make the chart clearer, we removed some countries and reduced it to the years from 2010 to 2013. Further we applied a min-max normalization.

In this experiment, we want to illustrate the differences of a clustering which received a good score and another clustering which received a worse one. Therefore we clustered the dataset with seeded KMeans and different k.

In Figure 5 it can be seen, that the clustering with $k = 8$ received a CLOSE score of 0.67, which represents the best score. The clustering itself can be seen in Figure 6. In order to compare this clustering to another with a lower score Figure 6 also holds the clustering result for $k = 3$.

In direct comparison the first differences that stand out are the cluster sizes. The clusters received with $k = 8$ are smaller than those of $k = 3$. This alone is no surprise but it leads to a smaller MSE and thus to a lower negative influence on the CLOSE score. In numbers, the average MSE for $k = 8$ is 0.0036. For $k = 3$ it is 0.0289. The second not so obvious observation is the average cluster stability. While the clustering with $k = 3$ has an average stability of 0.56, the agglomerations found with $k = 8$ got an average stability of 0.68. One example which leads to a higher stability is the behavior of the object BRB and its neighborhood. In the clustering with the highest CLOSE score, BRB has the same cluster neighbors in the first and the last year. In addition it is alone in a cluster in 2012, which means it moved with 50% of its neighbors from 2010. This is not the case in the clustering which was found with $k = 3$. In fact in the clustering with $k = 3$, BRB is never in a cluster with the same cluster members

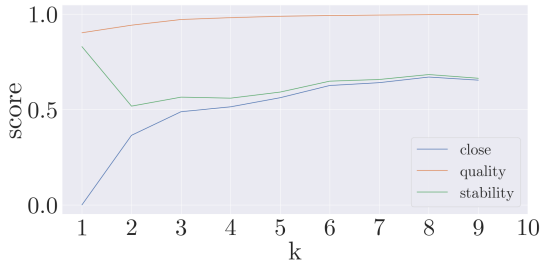


Fig. 5: Achieved scores for different k on the GlobalEconomy data set.

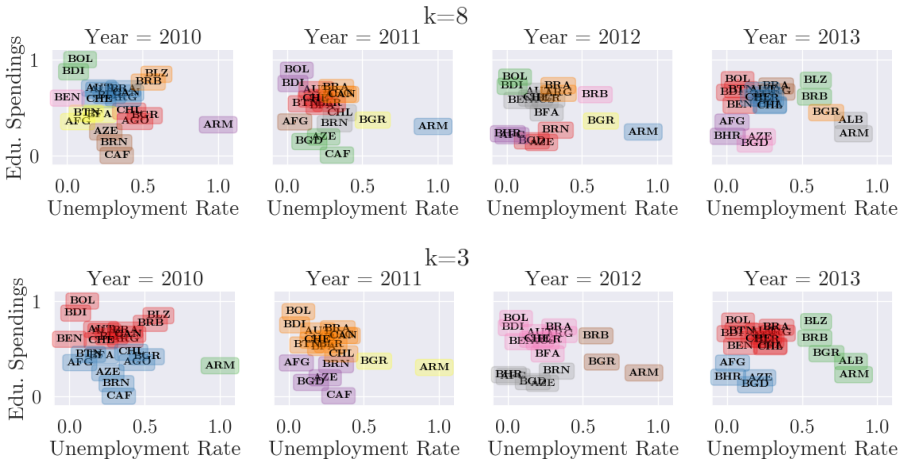
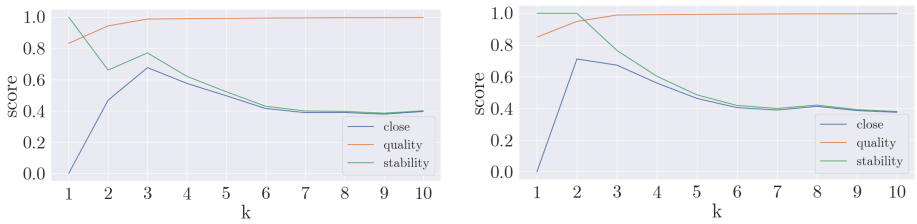


Fig. 6: KMeans Clusters with $k = 8$ and $k = 3$ on the GlobalEconomy data set. The datapoints contain ISO Countrycodes.

over two years. Another observation in the clustering with $k = 3$ is, that data points which change their cluster neighbors over time often move with a low number of other data points.

5.3 Artificially Generated Data Set

To show what a good clustering and the associated CLOSE score may look like, we generated two artificial data sets. In both cases, at first three random centroids with two features $\in [0, 1]$ were chosen. Then 20, 15 and 10 time series were placed next to these centroids, respectively. This means that the data points



(a) Data Set A (b) Data Set B

Fig. 7: Achieved CLOSE scores, average quality and average *ot_stability* for the two generated data sets depending on k. The quality line is given by 1 - MSE.

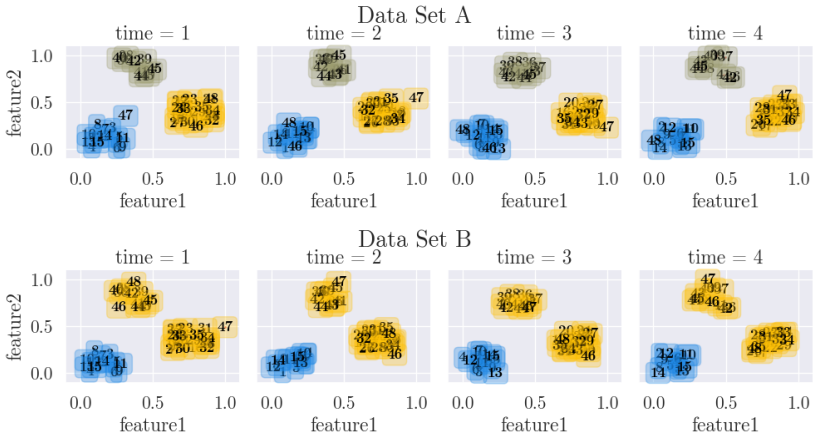


Fig. 8: Detected clusters by KMeans on the two artificially generated data sets.

of a time series for each time point were set with a maximal distance of 0.1 per dimension to the assigned centroid. Subsequently, data points for 3 time series (namely 46, 47 and 48) with random transitions between two of the three clusters were placed in the feature space. For overview purposes a total of 4 time points and 48 time series were examined. In Figure 8 the resulted data sets can be seen. Data set A contains transitions between the two lower clusters. In data set B there are transitions between the two upper clusters.

The clustering was performed with KMeans [17] for $1 \leq k \leq 10$. Figure 7 shows the achieved CLOSE scores, average quality and average *ot_stability* depending on k, whereby the quality line is given by 1 - MSE. While for data set A the best k is in accordance to the chosen centroids three, for data set B $k = 2$ is preferable. The corresponding clustering results are illustrated in Figure 8. The outcomes show that the best results regarding the CLOSE score may deviate from those of normal clustering if a fusion/split of clusters can increase the over-time stability without causing significant quality loss. As in data set B the clusters with bouncing time series are located close together, a merge of the two clusters is beneficial: the quality is only slightly affected, while the stability is significantly increased.

6 Discussion

Clustering time series is a challenging task. Besides the methodology, the user needs to choose parameters, which all lead to different results. Improving the results by adapting the parameters is often only possible with the help of a specialist. In this paper we provide a systematic approach for the determination of parameters in order to reach a given target. This enables users not only to

compare different clusterings, but also to choose a method and parameters suited for the data set without further knowledge.

Further more our work enables the user to use an arbitrary cluster algorithm and distance function, without further adaptation. If considering uniformly populated convex data groups, measures such as the mean squared error (MSE) or mean absolute error (MAE), and distance- or partition-based clustering algorithms such as KMeans are suitable. If the data set contains groups whose members are not approximately normally distributed, density-based measures such as the local outlier factor (LOF) and clustering algorithms such as DBSCAN might be more appropriate. Additionally, the formula of CLOSE (7) can be modified, so that quality measures for clusterings instead of clusters can be used. In that case, the average cluster stability avg_stab for every clustering ζ_{t_i} at time t_i can be considered:

$$CLOSE(\zeta) = \frac{1}{N_C} \cdot \Big(1 - \Big(\frac{k}{N_C}\Big)^2\Big) \cdot \Big(\sum_{\zeta_{t_i} \subset \zeta} avg\_stab(\zeta_{t_i}) \cdot (1 - quality(\zeta_{t_i}))\Big). \quad (8)$$

We are aware, that the presented method is computationally intensive but we are confident to enhance the approach in the future. Moreover, this is only a small drawback in view of the fact, that the complex manual search, which itself is very time-consuming anyway, gets simplified and guided.

7 Conclusion and Future Work

The presented method can be divided into two major parts: First the rating of time series and their subsequences, and second the evaluation of over-time clusterings. In this paper we focused on the latter. Therefore we presented a robust method which is able to rate over-time clusterings regarding a temporal linkage. This enables the comparison of different clusterings and their bond in time. We have performed several experiments and explained the influence of the major factors. The results show that our method is able to measure the over-time stability accurately for over-time clusterings of multiple multivariate time series. With the help of the presented measure, stable clusters are found. Due to the consideration of the quality, however, no unintuitive clusters are forced in favor of stability.

Based on CLOSE, much further research can be done. Apart from investigating different quality measures for clusterings, the treatment of outliers can be contextually adapted and analyzed. One way to penalize noise would be, to replace k_a in the subsequence score (3) with k. This would cause, that outliers would get the worst score of 0, as the timestamps would not be skipped. Besides, an intelligent initialization of the reference timestamp could be developed. Instead of examining the behavior with respect to the first point in time, e.g. the time with the highest clustering quality could be chosen. Furthermore, CLOSE can be used to detect anomalous subsequences using the *subsequence score* [25].

The presented measure could also be used in streaming environments. For example, it could indicate a significant change of data composition. Social media

could be an interesting field of application, too. The subsequence score of Instagram followers could e.g. be an indicator for their probability of remaining as a follower. In addition, the combination of CLOSE with contextual clustering [11] might lead to deeper insights about the resulting cluster compositions. Another interesting aspect would be the development of an over-time clustering algorithm using CLOSE as objective function. This would make the time-consuming search for optimal parameters per time point disappear.

Acknowledgement

We would like to thank the Jürgen Manchot Foundation, which supported this work by financing the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Duesseldorf.

References

1. Global economy, world economy, https://www.theglobaleconomy.com/, accessed: 13.01.2020
2. Banerjee, A., Ghosh, J.: Clickstream clustering using weighted longest common subsequences. In: Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining. pp. 33–40 (2001)
3. Ben-David, S., Von Luxburg, U.: Relating clustering stability to properties of cluster boundaries. In: 21st Annual Conference on Learning Theory (COLT 2008). pp. 379–390 (2008)
4. Chen, J.R.: Useful clustering outcomes from meaningful time series clustering. In: Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70. pp. 101–109 (2007)
5. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979)
6. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics **3**(3), 32–57 (1973)
7. Ernst, J., Nau, G.J., Bar-Joseph, Z.: Clustering short time series gene expression data. Bioinformatics **21**(suppl_1), i159–i168 (2005)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231 (1996)
9. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE Trans. on Knowl. and Data Eng. **15**(3), 515–528 (2003)
10. Huang, X., Ye, Y., Xiong, L., Lau, R.Y., Jiang, N., Wang, S.: Time series k-means: A new k-means type smooth subspace clustering for time series data. Information Sciences **367-368**, 1 – 13 (2016)
11. Jänichen, S., Perner, P.: Conceptual clustering and case generalization of two-dimensional forms. Computational Intelligence **22**(3-4), 177–193 (2006)
12. Jin, X., Lu, Y., Shi, C.: Distribution discovery: Local analysis of temporal rules. In: Advances in Knowledge Discovery and Data Mining. pp. 469–480 (2002)

13. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Advances in Spatial and Temporal Databases. pp. 364–381 (2005)
14. Keogh, E., Lin, J.: Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowledge and Information Systems **8**(2), 154–177 (2005)
15. Kuncheva, L.I., Vetrov, D.P.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1798–1808 (2006)
16. von Luxburg, U.: Clustering stability: An overview. Found. Trends Mach. Learn. **2**(3), 235–274 (2010)
17. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297 (1967)
18. O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Proceedings of IEEE International Conference on Data Engineering. p. 685 (2001)
19. Paparrizos, J., Gravano, L.: k-shape: Efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1855–1870 (2015)
20. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association **66**(336), 846–850 (1971)
21. Reuters, T.: Eikon financial analysis and trading software
22. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53 – 65 (1987)
23. Schäfer, P.: Experiencing the shotgun distance for time series analysis. Trans. MLDM **7**, 3–25 (2014)
24. Schäfer, P.: Towards time series classification without human preprocessing. In: MLDM 2014: Machine Learning and Data Mining in Pattern Recognition. pp. 228 – 242 (2014)
25. Tatusch, M., Klassen, G., Bravidor, M., Conrad, S.: Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In: Data Mining. AusDM 2019. Communications in Computer and Information Science. vol. 1127, pp. 91–103 (2019)
26. Truong, C.D., Anh, D.T.: A novel clustering-based method for time series motif discovery under time warping measure. International Journal of Data Science and Analytics **4**(2), 113–126 (2017)

Coarse- and fine-scale geometric information content of Multiclass Classification and implied Data-driven Intelligence.

Fushing Hsieh*[1] and Xiaodong Wang[1]

[1] University of California, Davis CA 95616, USA
{fhsieh, xidwang}@ucdavis.edu

Abstract. Under any Multiclass Classification (MCC) setting defined by a collection of labeled point-cloud specified by a feature-set, we extract only stochastic partial orderings from all possible triplets of point-cloud without explicitly measuring the three cloud-to-cloud distances. We demonstrate that such a collective of partial ordering can efficiently compute a label embedding tree geometry on the Label-space. This tree in turn gives rise to a predictive graph, or a network with precisely weighted linkages. Such two multiscale geometries are taken as the coarse scale information content of MCC. They indeed jointly shed lights on explainable knowledge on why and how labeling comes about and facilitates error-free prediction with potential multiple candidate labels supported by data. For revealing within-label heterogeneity, we further undergo labeling naturally found clusters within each point-cloud, and likewise derive multiscale geometry as its fine-scale information content contained in data. This fine-scale endeavor shows that our computational proposal is indeed scalable to a MCC setting having a large label-space. Overall the computed multiscale collective of data-driven patterns and knowledge will serve as a basis for constructing visible and explainable subject matter intelligence regarding the system of interest.

Keywords: Label embedding tree · Partial ordering · PITCHf/x.

1 Introduction

Nowadays Machine Learning (M.L.) based Artificial Intelligence (A.I.) researches are by-and-large charged to endow machines with various human's semantic categorizing capabilities [13]. Given that human experts hardly make semantic categorizing mistakes, should machine also help to explain: How and Why, to human? We demonstrate that possible answers are computational and visible under any Multiclass Classification (MCC) setting. The keys are: first compute the pertinent information content without artificial structure; secondly, graphically display such information content via multiscale geometries, such as a tree, a network or both, to concisely organize and deliver pattern-based knowledge or intelligence contained in data to human attentions.

Multiclass Classification is one major topic [2,3,5,9,16] of associating visual images or text articles with semantic concepts [7,12,17]. Its two popular techniques: flat and hierarchical, are prone to make mistakes [1,6,10]. Since a machine is primarily forced to assign a single candidate label toward a prediction. No less, no more. Such a forceful decision-making to a great extent ignores the available amount of information supported by data. With such kind of M.L. in the heart of A.I., it is beyond reasonable doubt that A.I. is bound to generate fundamental social and academic issues in the foreseeable future, if its error-prone propensity is not well harnessed in time.

If completely error-free A.I. is not possible at current state of technology, then at least it should tell us its decision-making trajectory leading up to every right or wrong decision. It is in the same sense as the recommended fourth rule of robotics: "a robot or any intelligent machine-must be able to explain itself to humans" to be added to Asimov's famous three. Since we need to see why, how and where errors occur in hope of knowing what causes, and even figuring out how to fix it.

Such a quality prerequisite on A.I. and M.L. is also coherent with concurrent requirements put forth by many governments around the world: Transparent explanation upon each A.I. based decision is required. Now it is a critical time point to think about how to coherently build and display data's authentic information content that can afford the making of explainable error-free decisions. So such information content with pertinent graphic display can be turned into Data-driven Intelligence. In this paper, we specifically demonstrate Data-driven Intelligence for Multiclass Classification. This choice of M.L. topic is in part due to that classification is human's primary way of acquiring intelligence, and also in part due to its fundamental importance in science and industry.

On the road to Data-driven Intelligence, we begin by asking the following three simple questions. First, the naive one is: Where is relevant information in data? Secondly, what metric geometry is suitable to represent such information content? Finally, how to make perfect, or at least nearly perfect empirical inference or predictive decision-making? We address these three non-hypothetical questions thoroughly based on model-free unsupervised M.L. Here we explicitly show the nature of information content under Multiclass Classification as: multiscale heterogeneity. Such information heterogeneity can be rather intertwined and opaque when its three data-scales: numbers of label, feature and instance, are all big.

The paper is organized as follows. In section 2 we describe the background and related work of MCC. In section 3 we develop a new label-embedding tree constructed via partial ordering and a classification schedule. In section 4 we illustrate a tree-decent procedure with early stop and represent the error flow. In section 5 we explore the heterogeneity embedded within labels.

2 Multiclass Classification

A generic Multiclass Classification (MCC) setting has three data scales: the number of label L, the number of feature K and total number of subjects N. Each label specifies a data-cloud. A data-cloud is an ensemble of subjects. Each subject is identified by a vector of K feature measurements. The complexity of data and its information content under any MCC setting is critically subject to L, K and N. The goal of Multiclass Classification is to seek for the principles or intelligence that can explain label-to-feature linkages. Such linkages are intrinsically heterogeneous as being blurred by varying degrees of mixing among diverse groups within the space of labeled data-clouds. Since such data mixing patterns are likely rather convoluted and intertwined, so the overall complexity of information content must be multiscale in nature.

Specifically speaking, its global scale is referred to which label's point-cloud is close to which, but far away from which. Though such an idea of closeness is clearly and fundamentally relative, it is very difficult to define or evaluate precisely. That is, such relativity essence can't be directly measured with the presence of two point-clouds, but it can be somehow reflected only in settings involving three or more point-clouds. From this perspective, all existing distance measures commonly suffer from missing the data-clouds' essential senses of relative closeness locally and globally. For instance, recently Gromov-Wasserstein distance via Optimal Transport has been proposed as a direct evaluation of distance between two point-clouds [14]. But it suffers from the known difficulty in handling high dimensionality (large K). So this distance measure likely misses the proper senses of relative closeness among point-clouds, especially when K is big.

In this paper, we propose a simple computing approach to capture the relative closeness among all involving point-clouds without directly and explicitly evaluating pairwise cloud-to-cloud distance. The key idea is visible as follows: through randomly sampling a triplet of singletons from any triplet of point-clouds, we extract three partial ordering among the three pairs of cloud-to-cloud closeness. By taking one partial ordering as one win-and-loss in a tournament involving $\binom{L}{2}$ teams, we can build a dominance matrix that leads to a natural label embedding tree as a manifestation of heterogeneity on the global scale. Such a triplet-based brick-by-brick construction for piecing together a label embedding tree seems intuitive and natural. Indeed such a model-free approach is brand new to M.L. literature [3, 4]. The existing hierarchical methods build a somehow symbolic label embedding tree by employing a bifurcating scheme that nearly completely ignores the notion of heterogeneity [3, 11, 15].

After building a label embedding tree on the space of L labels, we further derive a predictive graph, which is a weighted network with precisely evaluated linkages. This graph offers the detailed closeness from the predictive perspective as another key aspect of geometric information content of MCC.

To further discover the fine scale information content of MCC, we look into heterogeneity embraced by each label. Clustering analysis is applied on each label's point-cloud to bring out a natural clustering composition, and then label

each cluster pertaining to a sublabel. By doing so across all labels, we result in a space of sublabel with much larger size than L. Likewise we compute a sublabel embedding tree and its corresponding predictive graph. These two geometries then constitute and represent the fine scale information content of MCC.

A real database, Major League Baseball (MLB) $PITCHf/x$, is analyzed for the purpose of application. Since 2008 the PITCHf/x database of MLB has been recording each every single pitch delivered by MLB pitchers in all games at its 30 stadiums. A record of a pitch is a measurement vector of 21 features. A healthy MLB pitcher typically pitches around 3000 pitches, which are algorithmically categorized into one of pitch-types: Fastball, Slider, Change-up, curveball and others types.

We collect data from 14 ($= L$) MLB pitchers, who threw around 1000 Fastball or more during the 2017 season. As one pitcher is taken as a label, his seasonal fastball collection is a point-cloud. It is noted that each pitcher tunes his Fastball slightly and distinctively when facing different batters under different circumstances of game. That is, multi-scale heterogeneity is inherently embedded into each point-cloud.

A potential feature set is selected based on permutation-based feature importance measure. The importance score is defined as the reduction in the performance of Random Forest after permuting the feature values. All real data illustrations for the entire computational developments throughout this paper is done with respect to a feature set consisting of 3 features: horizontal and vertical coordinates, and horizontal speed of a pitch at the releasing point. Results on two larger feature-set are also reported.

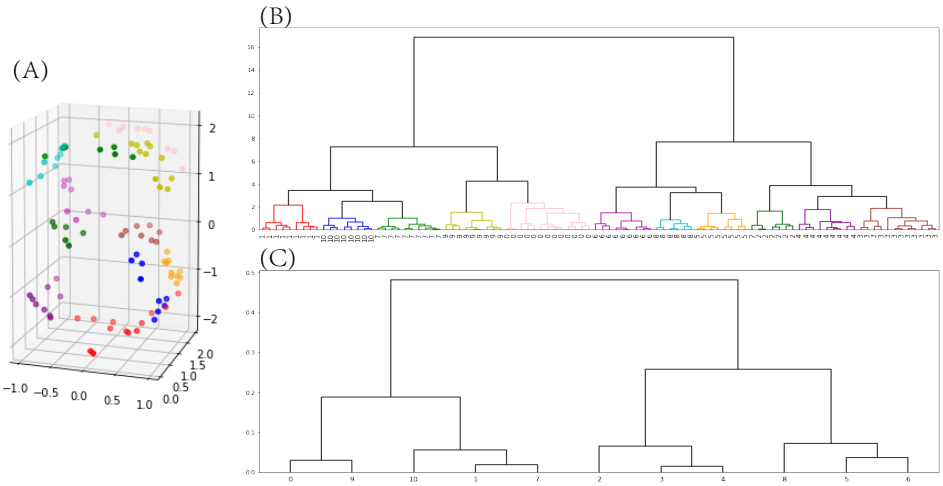


Fig. 1. Illustrating example for the Algorithm of label embedding tree. (A) the 3D scatter plot of data; (B) the 11 labeled data-clouds defined by a tree; (C) the embedding tree.

3 A label embedding tree built by partial ordering.

We develop a computing paradigm based on unsupervised machine learning to nonparametrically construct the label- and sub-label embedding trees in this paper. This paradigm is designed to be scalable to the three factors: L, K and N. With a label-triplet, say (La, Lb, Lc), in the brick-by-brick construction, partial ordinal relations are referred to: $D(La, Lb) < D(La, Lc)$ for example, where $D(.,.)$ is the unspecified "distance" between two label clouds. **It is emphasized that the Alg.1 is devised to extract such relations without explicitly computing the three pairwise distances** $D(.,.)$. These relations found among three point-clouds are stochastic in nature.

Given a triplet of labels La, Lb, Lc, if we randomly sample three singleton vectors in R^K, say X_{La}, X_{Lb} and X_{Lc}: one from each of three labels, separately. A piece of information of partial ordering within the triplet can be shed by inequalities among Euclidian distances $d(.,.)$ among 3 singletons X_{La}, X_{Lb} and X_{Lc}. That is, inequality $d(x_{La}, x_{Lb}) < d(x_{La}, x_{Lc})$ provides a small piece of information about Labels La and Lb being closer than La to Lc and Lb to Lc. By iteratively randomly sampling vector-triplets for a large number of times, say T, the probability of this relative closeness between La and Lb can be estimated as $\hat{P}(D(La, Lb) < D(La, Lc)) = \sum_t 1_{D(La,Lb)<D(La,Lc)}/T$. Via Law of Large number, we arrive at the relative closeness information by aggregating partial ordering among all possible combination of three labels. Let H be a square dominant matrix with $\binom{L}{2} = L(L-1)/2$ rows and columns. Each entry of H records a probability that "this unspecified distance $D(.,.)$ of a label-pair" is dominated by the same unspecified distance of another label-pair. Denote i_{xy} is the index of a label pair Lx and Ly. The entry of H in the i_{ab}th row and the i_{cd}th column records the related probability between these two label pairs.

$$H[i_{ab}, i_{cd}] = P(D(La, Lb) < D(Lc, Ld)) \tag{1}$$

It is noted that $H(i, j) + H(j, i)$ is equal to 1. In this way, H realizes the partial ordering among all pairs of labels.

It is worth noting that such a dissimilarity matrix $\bar{D}$ is by no means a metric satisfying triangular inequality or other properties. Here we illustrate the validity of this algorithm through a small example, as shown in Fig.1. A S-shape data set is simulated in R^3 space, see panel (A). Hierarchical clustering is implemented and a dendrogram is shown in panel (B). 11 clusters are obtained by cutting the dendrogram at a certain tree height and each cluster is marked with different color. Consider each cluster as a label, and a label embedding tree is created via Algorithm1 to show the hierarchical structure among those 11 classes in (C). It shows that our labeling tree built by only using partial ordering can reflect the original hierarchy among labels very well. In short, our dissimilarity matrix makes more sense in showing the natural label-cloud hierarchical dependency, which is the most advantage to distinguish our labeling tree from others.

There is a natural way to do classification based on this triplet partial ordering. We can simply assign a singleton or a batch of unlabeled sample with a new

label L_{new}, which never appears in the previous label set. So there is supposed to be $L+1$ labels in total. Then, the triplet-version dissimilarity $\binom{L+1}{2} \times \binom{L+1}{2}$ matrix H_{new} can be calculated for all those $L+1$ labels. The classified label is just the one that is the closest to the new label, see **Alg.2**. Actually, given the previous $\binom{L}{2} \times \binom{L}{2}$ matrix H pre-trained, it is only necessary to calculate the rest $\binom{L+1}{2} \times L$ sub-matrix. That is to say, we randomly sample two singletons X_{La} and X_{Lb} from two labels La and Lb, respectively, and sample one unlabeled sample X_{new} from L_{new}. The partial ordering now turns out to compare $d(X_{La}, X_{new})$ and $d(X_{Lb}, X_{new})$. Via a large number of sampling, we gain information about $P(D(La, L_{new}) < D(Lb, L_{new}))$ and its counterpart. Let H_{new} record all newly added probabilities of such dominance. Then the label-pairwised dissimilarity matrix is calculated via the column sum of H_{new}. Therefore, the classification procedure is equivalent to aggregating all binary classifiers and vote according to the sum of probability, which is exactly one-versus-one classification with a soft vote strategy. One brand new property is that, when L_{new} represents a unlabeled data-cloud, the geometry of this data-cloud is fully used in this predictive decision-making.

Alg.1 Label Embedding Tree

Denote: H is a $\binom{L}{2} \times \binom{L}{2}$ ranking matrix,
$H[i_{ab}, i_{cd}] = P(D(La, Lb) < D(Lc, Ld))$
where i_{ab} is the index of label pair La and Lb, $D(La, Lb)$ is their dissimilarity which is inaccessible.
Initialize: H with all entries 0
for (La, Lb, Lc) in all unique label triplets:
Randomly sampling a triplet of data for T times with replacement, denoted as $(X_{La}^{(1)}, X_{Lb}^{(1)}, X_{Lc}^{(1)})$, $(X_{La}^{(2)}, X_{Lb}^{(2)}, X_{Lc}^{(2)})$, ..., $(X_{La}^{(T)}, X_{Lb}^{(T)}, X_{Lc}^{(T)})$
where X_L is a single sample of data with label $y = L$
for t in $1, ..., T$:
if $d(X_{La}^{(t)}, X_{Lb}^{(t)}) < d(X_{La}^{(t)}, X_{Lc}^{(t)})$: $H[i_{ab}, i_{ac}] += 1/T$
else $H[i_{ac}, i_{ab}] += 1/T$
if $d(X_{La}^{(t)}, X_{Lb}^{(t)}) < d(X_{Lb}^{(t)}, X_{Lc}^{(t)})$: $H[i_{ab}, i_{bc}] += 1/T$
else $H[i_{bc}, i_{ab}] += 1/T$
if $d(X_{La}^{(t)}, X_{Lc}^{(t)}) < d(X_{Lb}^{(t)}, X_{Lc}^{(t)})$: $H[i_{ac}, i_{bc}] += 1/T$
else $H[i_{bc}, i_{ac}] += 1/T$
end for
end for
Calculate $K \times K$ labeling dissimilarity matrix $\bar{D}$
$\bar{D}(La, Lb) = E_{Lx,Ly}\{P(D(Lx, Ly) < D(La, Lb))\} = \sum_j H(j, i_{ab}) / \binom{L}{2}$
Output: a hierarchical clustering tree based on the dissimilarity matrix $\bar{D}$

We can also sample X_{La} and X_{Lb} from the neighbors of X_{new} to extract the partial ordering locally. Let's choose M-nearest neighbors of X_{new} constrained in the data with label La, denoted as $X_{M|La} = (X_{La}^{(1)}, X_{La}^{(2)}, ..., X_{La}^{(M)})$, and so is $X_{M|Lb}$. We look at whether there are relatively more La's compared with Lb's in the M nearest neighbors. This classification becomes k-Nearest Neighbor with tuning parameter k chosen to be M. If we repeat the aforementioned procedure for a large number of times, we have another way of extracting information of $P(D(La, L_{new}) < D(Lb, L_{new}))$. Thus, Algorithm2 is equivalent to one-versus-one classification with k-NN as its classifier. These properties also explain why our triplet comparison is so important.

Besides, Algorithm2 can indicate where the unknown label is located within the previous label embedding tree. The label embedding tree with an unknown label embedded is clear to view which labels are mixed with the unknown label in a small branch and which labels is far away. See Fig.2 for an illustration.

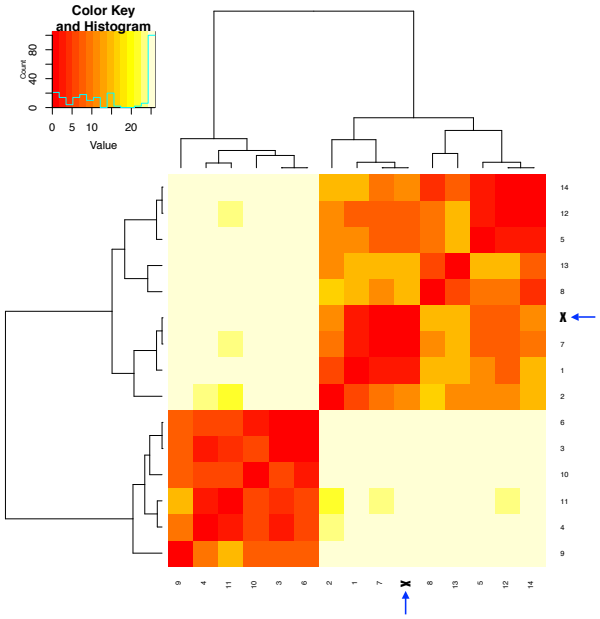


Fig. 2. Label embedding tree of 14 pitchers with a heatmap of "distance" derived from a computed H and an illustrating example of classifying an unknown label **X**; the ground truth is label 7

The number of sampling iteration T is supposed to be as large as possible. In practice, T should be chosen dependent on the sample size of each label. If the data is balanced, $T = N/L$, otherwise, $T = max_i N_i$ to cover the biggest label data cloud, where N_i is the sample size for label i. So the time complexity is $O(NKL^2)$.

When L is small or moderate, consider a setting with the number of all possible triplets, $\binom{L}{3}$, being not overwhelmingly big. We perform Algorithm1 on all possible triplets to fill up the $\binom{L}{2} \times \binom{L}{2}$ dominance matrix, H. Each of column sum of H tells how many times a label-pair's distance is dominated by distances of all other pairs. So the bigger a column sum is, the larger degree of similarity of this label pair is. Therefore the $\binom{L}{2}$-vector of column sums of H can be transformed into a natural $L \times L$ similarity matrix, say $\bar{S}$, among all involving labels. In contrast, the $\binom{L}{2}$-vector of row sums of H is a distance (dissimilarity) matrix, say $\bar{D}$, of all labels. Such a $\bar{S}$ or $\bar{D}$ will afford a hierarchy, which is the label embedding tree.

Alg.2: Classify X_{new} with an unknown label Lx

Input: a $\binom{L}{2} \times \binom{L}{2}$ matrix H obtained from **Alg.1**
Initialize: a $\binom{L+1}{2} \times \binom{L+1}{2}$ ranking matrix H_{new}
$H_{new}[1 : \binom{L}{2}, 1 : \binom{L}{2}] = H$ and the rest entry 0.
for (La, Lb) in all unique label pairs:
    Randomly sampling a pair of data for T times with replacement, and concatenate it with X_{new} to make a triplet, denoted as $(X_{La}^{(1)}, X_{Lb}^{(1)}, X_{new})$, $(X_{La}^{(2)}, X_{Lb}^{(2)}, X_{new})$, ..., $(X_{La}^{(T)}, X_{Lb}^{(T)}, X_{new})$
    where X_L is a single sample of data with label $y = L$
    for t in $1, ..., T$:
        if $d(X_{La}^{(t)}, X_{new}) < d(X_{Lb}^{(t)}, X_{new})$, $H_{new}[i_{ax}, i_{bx}] += 1/T$
        else $H_{new}[i_{bx}, i_{ax}] += 1/T$
        where i_{ax} and i_{bx} are indices for label pair (La, Lx) and (Lb, Lx)
    end for
end for
Output1: Classification result a^*, if
    $i^* = i_{a^*x}$, $i^* = \text{argmin}_i \sum_j H(j, i)$
Get $(L+1) \times (L+1)$ dissimilarity matrix $\bar{D}_{new}$
$\bar{D}_{new}(La, Lb) = \sum_j H(j, i_{ab}) / \binom{L+1}{2}$
Output2: a hierarchical clustering tree on $\bar{D}_{new}$
The tree can also return in which branch the unknown label Lx locates from the previous labeling tree.

When it is too expansive to compute a full version of H, then we start with a sparse version, says H'. By applying the transitivity property in dominance relationship, we can resolve the sparsity issue by making product matrix like $H = H' \times H'$ to record all indirect dominance with one intermediate [8], see the Algorithm-A in Appendix. By embracing such transitivity, as confirmed in our experiment, a reliable distance dominance matrix H can be resulted.

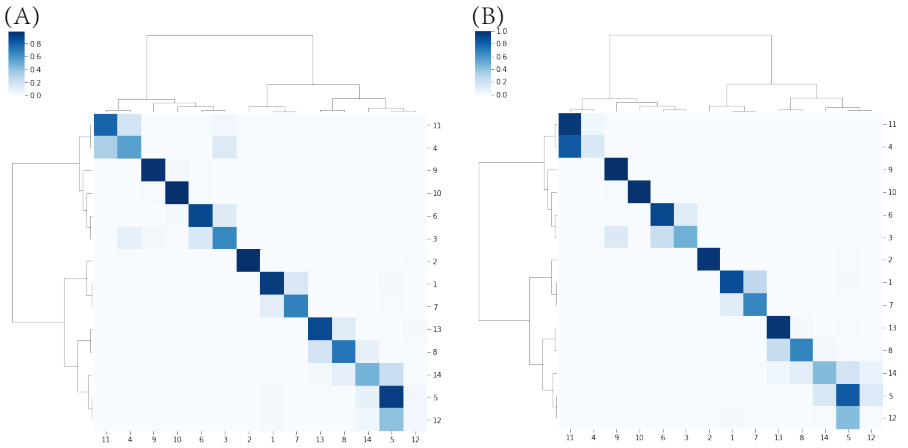


Fig. 3. Label embedding tree superimposed on its confusion matrix: (A) Classification being driven to the tree bottom with a singleton label candidate; (B) Classification can stop early at a tree inter-node.

4 Tree-descent schedule and error flow

With a label embedding tree, a very efficient decision-making process can be devised via tree descend framework as depicted in **Alg.3**. This algorithm works for any bi-class classifier by making a chain of decisions from top-to-bottom levels of the label embedding tree. So our label embedding tree becomes a scalable platform for decision-making with respect to the number of labels (L). In fact the tree somehow provides an ideal setting for distance metric learning [18], because similar labels have been clustered together.

For prediction purpose, ideally the tree's binary branching structure can allow us to arrive at a singleton label at the bottom of tree, or a small set of label as a small tree branch by avoiding any risk of making any major mistake. A threshold θ defined in Algorithm3 works for risk control. If the probability of classification is less than θ, say 0.8, we have less confidence to descend the labeling tree further, so early stop the iteration and return a label set. This fact can be visualized from our construction of predictive graph below.

Alg.3 Classify X_{new} via descending label embedded tree with an early stop

---

Input: a label embedding tree B; a trained Binary Classifier F; a threshold θ to stop descending tree

Denote:

B_{Left} and B_{Right} are the left and right branch on the root node of tree B

$F_L(X_{new})$ returns the probability of classifying X_{new} into Left branch

$F_R(X_{new})$ returns the probability of classifying X_{new} into Right branch

while $(|B| > 1$ & $max\{F_L(X_{new}), F_R(X_{new})\} > \theta)$:
if $F_L(X_{new}) > F_R(X_{new})$, **then** $B \leftarrow B_{Left}$
else $B \leftarrow B_{Right}$
end while
Output: label(s) under the current tree B

Let $\mathcal{Y} = \{L_j\}_1^L$ and $\mathcal{F} = \{f_i\}_1^K$ be the ensembles of label and feature, respectively. Denote a computed label embedding tree as $\mathcal{B}[\mathcal{F}]$. We derive a label predictive graph, denoted by $\mathcal{G}[\mathcal{F}]$, based on a confusion matrix. All classification results are collectively summarized into an asymmetric error-flow matrix $\mathcal{E}[\mathcal{F}] = [e_{i,j}]$ with directed error-flows $(e_{i,j}, e_{j,i})$ between any label pair (L_i, L_j) are the percentages of wrong decisions by predicting L_i to be L_j, and vice versa. $\mathcal{G}[\mathcal{F}]$ is a weighted network or graphic representation of $\mathcal{E}[\mathcal{F}]$, see Fig.3 for two predictive graphs of 14 MLB pitcher-labels.

The essence of $\mathcal{G}[\mathcal{F}]$ is that its pairwise directional links $\{(e_{i,j}, e_{j,i})\}$ realistically reflects unequal mixing configurations of labels L_i from L_j. The utility of $\mathcal{G}[\mathcal{F}]$ is that it allows a smallest predictive label set, while achieving a nearly perfect precision. Such an asymmetry, See Fig.3, is invaluable in understanding the MCC setting and in explaining decision-making. This perspective is completely lost when a direct distance measure is forcefully employed.

With the two explicit and visible geometries embraced by the computed label embedding tree and its corresponding predictive graph as the MCC information content with respect to feature set $\mathcal{F}$, the linkages between the label space $\mathcal{Y}$ and the collection of point-clouds defined by feature set $\mathcal{F}$ become evidently explainable. It is clear to see that the predictive graph is possible to guide us to error-free decision-making if our decision is in a form of a set of potential label candidates, rather than restricted to a singleton. This fact leads us to reflect on the common phenomenal issue: Why predicting an unlabeled singleton has to be prone to error? There are at least two key reasons. First, a predictive object can be caught deep within some point-clouds of wrong labels, not just the right one. Therefore, involving all labels' data-clouds at once for such prediction is not ideal. To ameliorate such a situation, a decision-making process descents from the top of a label embedding tree is strategic since MCC's information content is fully used. The second reason is that we ignore what amount of information is available, and simultaneously force ourselves to make a single pick of label.

5 Fine scale information content of MCC.

It is known that each label's point-cloud contains its own label specific heterogeneity. Discovering and accommodating such heterogeneity into MCC's information content in a collective fashion is another essential part of our data-driven computational endeavors. Since our label embedding tree can represent the natural hierarchical structure among separated data clouds, it is straightforward for us to decompose one label's point cloud into separate sublabel clusters and then

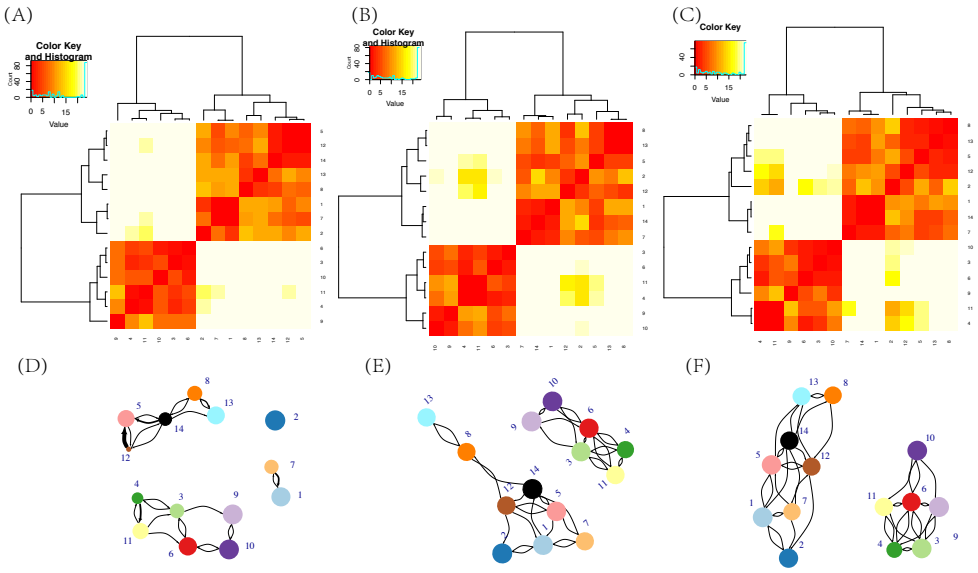


Fig. 4. dissimilarity matrix and predictive graphs calculated on 3 different Feature Groups with increasing sizes (see Group 1, 3 and 4 in Appendix)). (A),(B),(C) illustrate the dissimilarity matrix with a label embedding tree embedded on the row and column axis. The label number is the index of a baseball pitcher. There are 14 different pitcher, labeled from 1 to 14; (D),(E),(F) are predictive graphs that visualize the bi-class cut tree descending result.

implement Algorithm1. The sublabel clusters are empirically discovered from each label through a hierarchical clustering tree built upon this label's point could. On the MLB pitching MCC setting, 139 sublabels are generated. We then likewise construct a sublabel embedding tree and its corresponding 139×139 confusion matrix.

Both geometries of fine scale MCC's information content are shown in the three panels of Fig.5. They explicitly reveal detail and complex mixing patterns among the 139 sublabel specific point-clouds. Such fine scale information to a great degree reflect the coarse scale information, but at the same time shed many new lights on its own. For instance, we see how diverse subtypes are belonging to a pitcher's fastball. If all his subtypes are located in a relative small branch of the sublabel embedding tree, then this pitcher fastball pitches are rather uniform. In contrast, if his subtypes are located across several far apart branches, then this pitcher's fastball pitches are difficult to predict. Further we examine in explicit detail how his subtypes are mixing with other pitchers' via a predictive graph. Such examinations allow us to discover how and why this pitcher is in common with which pitchers, and how and why he is distinct with which pitchers. That is, these two geometries are platforms for discovering and establishing many ways of comparing MLB pitchers from many aspects. All these discoveries as diverse

parts of the collective knowledge made possible by the fine scale of information content of MCC.

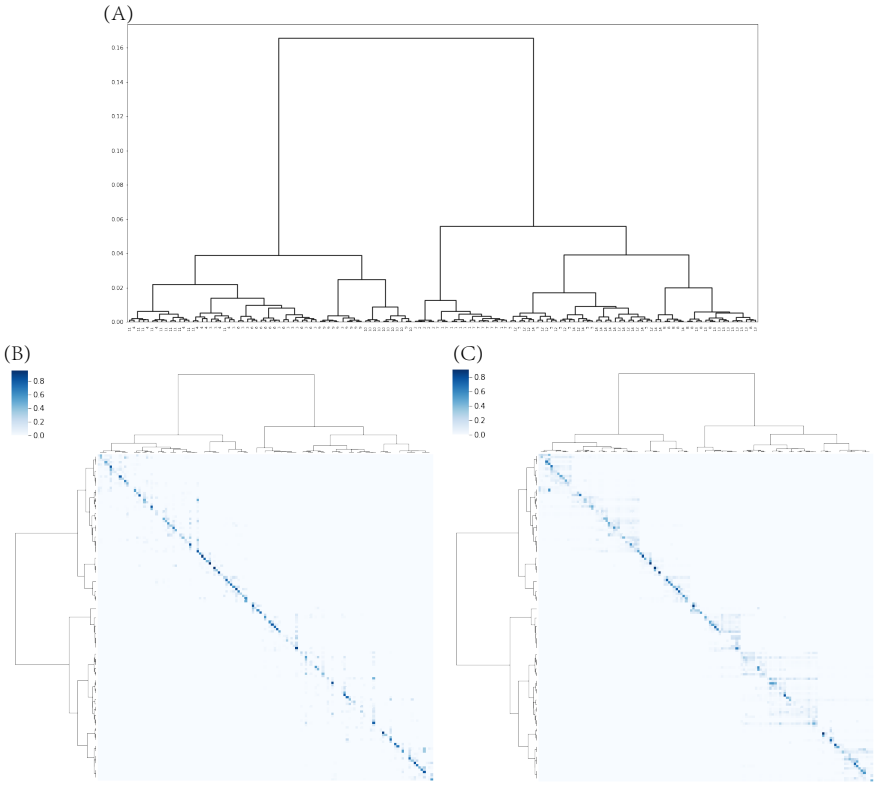


Fig. 5. Fine scale multiscale geometry of 139 sublabels, which belong to 14 pitchers labeled from 1 to 14. (A)The sublabel embedding tree; (B)the confusion matrix with a singleton label candidate; (C) predictions stop early at a tree inter-node.

6 Conclusion

The coarse and fine scales of information contents of MCC afford us to zoom-in and zoom-out to discover Data-driven Intelligence (D.I.) in visible and explainable fashion. The implied nearly perfect decision-making allows researchers to be responsible. We hope such a D.I. mindset can prevail from sciences to health industries, and beyond. Promoting D.I. is same as promoting truth and knowl-

edge already contained in data. Human might have been very wasteful in casting away invaluable knowledge by only focusing on forceful prediction.

Finally we make a remark on feature selection. Our standpoint here is that perfect decision-making is the prerequisite on any prediction issue occurring in sciences and health industries. Over these fields, any prediction needs to rightly reflect the amount of information available from data. At the same time, all decision-makers have to be responsible on what they decide. Their subject-matter sensitive criteria can be easily based on the two geometries of MCC's information content. That is, the task of feature selection shall be based on the $\mathcal{F}$ and be subject-matter sensitive. Such a standpoint is illustrated in Fig.4. By comparing the three sets of geometric information contents pertaining to three feature-sets (feature information given in Appendix), we gain different understanding and knowledge regarding the 14 MLB pitchers. We explain such Data-driven Intelligence(D.I.) pertaining to different sets of feature. That is why a prediction is better feature-set sensitive.

In summary, at least under MCC settings, Data-driven Intelligence is one basic principle objective of machine learning in Data Science as well as in Artificial Intelligence.

References

1. Allwein, E., Schapire, R., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. In: Journal of Machine Learning Research, vol. 1, pp. 113–141 (2000)
2. Amit, Y., Fink, M., Srebro, N., Ullman, S.: Uncovering shared structures in multiclass classification. In: Proceedings of the Twenty-fourth International Conference on Machine Learning (2007)
3. Bengio, S., Weston, J., Grangier, D.: Label embedding trees for large multi-class tasks. In: Advances in Neural Information Processing Systems 23, pp. 163–171 (2010)
4. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: Advances in Neural Information Processing Systems 28, pp. 730–738 (2015)
5. Cisse, M., Usunier, N., Artières, T., Gallinari, P.: Robust bloom filters for large multilabel classification tasks. In: Advances in Neural Information Processing Systems 26, pp. 1851–1859 (2013)
6. Cisse, M.: Efficient extreme classification. data structures and algorithms. In: cs.DS. Université Pierre et Marie Curie - Paris VI (2014)
7. Deng, J., Berg, A., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: K., D., P., M., N., P. (eds.) Computer Vision - ECCV 2010, vol. 6315. Springer, Berlin, Heidelberg (2010)
8. Fushing, H., Fujii, K.: Mimicking directed binary network for exploring systemic sensitivity: Is ncaa fbs a fragile competition system. In: Frontiers, Applied Mathematics and Statistics. (2016)
9. Gupta, M., Bengio, S., Weston, J.: Training highly multiclass classifiers. In: Journal of Machine Learning Research, vol. 15, pp. 1461–1492 (2014)
10. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: The Annals of Statistics, vol. 26, pp. 451–471 (2001)

11. Kosmopoulos, A., Partalas, I., Gaussier, E., Paliouras, G., Androutsopoulos, I.: Evaluation measures for hierarchical classification: a unified view and novel approaches. In: Data Mining and Knowledge Discovery, vol. 29, pp. 820–865 (2015)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Intelligent Signal Processing, pp. 306–351. IEEE (2001)
13. Russell, S.J., Norvig, P.: In: Artificial Intelligence: A Modern Approach (3nd ed.). Upper Saddle River, New Jersey: Prentice Hall. (2009)
14. Solomon, J.: Optimal transport on discrete domains. In: AMS Short Course on Discrete Differential Geometry. (2018)
15. Sun, A., Lim, E.: Hierarchical text classification and evaluation. In: Proceedings of the IEEE international conference on data mining, pp. 521–528 (2001)
16. Tsoumakas, G., Katakis, I., Vlahavas, I.: A review of multi-label classification methods. In: Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery, pp. 99–109 (2006)
17. Weinberger, K., Chapelle, O.: Large margin taxonomy embedding for document categorization. In: Advances in Neural Information Processing Systems 21, pp. 1737–1744 (2009)
18. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Journal of Machine Learning Research, vol. 10, pp. 207–244 (2009)

Appendix

Data Accessibility:

The pitching data is available in PITCHf/x database belonging to Major League Baseball via http://gd2.mlb.com/components/game/mlb/.

Feature explanation from PITCHf/x

A pitched baseball flight captured by 20 pairs of images via a pair of 60Hz cameras, which have orthogonal optical axes and cover the field of view between pitcher's mound and home plate, are determined with respect to the field coordinates. These images and estimated coordinates are converted into 21 features to characterize the flight's aerodynamics.

The 21 features are briefly described as follows:

1. The starting speed ("start speed") is measured when the ball is at the point 50 fts away from the home plate, which is very close to the horizontal and vertical coordinates of release point $(x0, z0)$ of a pitch.

2. The spin direction ("spin dir") is determined by assuming spin-axis being perpendicular to the movement direction, while spin rate ("spin rate") is the number of rotations per minute.

3. Vertical and horizontal movement measurements, denoted by "pfx-z" and "pfx-x", respectively. Topspin and backspin cause positive and negative vertical movements "pfx-z". Therefore this feature has a high association with "start speed" for pitchers, who has the high speed fastball as his chief pitch-type in his repertoire, than for pitchers, who doesn't. The feature "pfx-z" is also associated with features related to how a baseball trajectory curves.

4. A baseball trajectory from release point to the home plate is coupled with two straight lines: the tangent line at the release point (x_0, z_0) and the line links the release point and the trajectory's end point. The angle between these two lines is termed "break angle", while the maximum distance between the baseball trajectory and the second straight line is called and denoted as "break length". Therefore the three features: "pfx-z", "break angle" and "break length", are highly associated with each other.

5. The remaining features are three directions of speeds and accelerations at the release point, named "vx0, vy0,vz0" and "ax, ay, az", respectively, or play only auxiliary roles, like "break y", "x", and "y".

Feature Group1: "x0", "z0", and "vx0"

Feature Group2: "x0", "z0", "vx0", "vy0", "start-speed", "end-speed", and "spin-dir"

Feature Group3: "x0", "z0", "vx0", "vy0", "start-speed" "end-speed", "spin-dir", "spin-rate", "break-angle", "pfx-x", and "pfx-z"

Feature Group4: all 21 features

Alg.A Label Embedding Tree (Sparse)

Alg.1 is applied to get the dominance matrix $H^{'}$ with a smaller sampling iteration T
$H = H^{'} + H^{'} \times H^{'}$
$H(i,j) = min\{H(i,j), 1\}$
$\hat{D}(La, Lb) = \sum_j H(j, i_{ab}) / \binom{L}{2}$
Output: a label embedding tree based on $\hat{D}$

A New Ensemble Method for Convolutional Neural Networks and its Application to Plant Disease Detection

Vaibhav Kumar Katturu[1], ParamPuneet Kaur Thind[1], Teryn Cha[2], and Sung-Hyuk Cha[1]

[1] Computer Science Department, Pace University, New York, NY, USA
{vk38221n,pt54854n,scha}@pace.edu
[2] Computer Sciences, Essex County College, Newark, NJ, USA
yan@essex.edu

Abstract. Ensemble methods are of great importance in machine learning and data mining because combining multiple classifiers often results in better accuracy and predictions in many classification problems. Recently, ensemble methods with multiple Convolutional neural networks (CNN) have been suggested in various fields. Simple majority and conventional Borda voting methods were used in the ensemble CNN model. Here, a new voting method to combine multiple CNNs is proposed. The plant disease dataset containing five plant diseases that infect various plants is used to demonstrate the superiority of the proposed model. A total of 20 models to recognize five plant diseases were first built with different architectures. Experimental results of amalgamating CNNs with a new voting method suggest the superiority of the proposed voting method over other conventional methods.

Keywords: Borda Method · Convolutional Neural Network · Ensemble · Disease Detection · Voting

1 Introduction

A convolutional neural network, or simply CNN, is a Deep learning algorithm which can take in an image as an input, assign importance to various aspects or objects in the image, and be able to differentiate one from the other. A CNN is composed of multiple building blocks such as convolutional layers, poling layers and fully connected layers and is designed to automatically and adaptively learn spatial hierarchies of features through back propagation algorithm [14].

CNN has shown remarkable performance in computer vision tasks such as static image (object recognition) and videos (action or motion recognition) [10]. CNN has proven its worth in medical applications as well. In all these applications, CNNs are used to reach near precise accuracy though in some of the cases CNN doesnt perform well, due to (1) overfitting data while training and (2) compromised quality of images. Poor quality of images might leave out the region of interest. Both of these issues might lead to incorrect classification.

Consensus of a group plays an important role in decision making such as elections [7] and combining multiple classifiers [5]. It is essential in most democratic societies and has received great attention in artificial intelligence and computer science communities as well. Extensive surveys of preferential voting methods can be found in [1, 2]. Widely used conventional voting methods are simple majority voting if the top choice is selected and Borda method if all candidates are ranked. In this paper, modified Borda method is presented. Instead of the linear weights in the traditional Borda method, various weight vectors are tested. We claim that the power weight vector provides better results.

So as to demonstrate the superiority of the proposed voting method, plaint disease dataset which is one of the computer vision applications is used. To achieve accurate plant disease diagnostics a plant pathologist requires accurate observation skills so that one can identify characteristic symptoms. Variations in symptoms indicated by diseased plants may lead to incorrect diagnosis. Modern techniques for plant protection measures are required for effective control of diseases and undoubtedly been adopted by several researchers. For a decade now, it has been a practice to develop a model using CNN and train the model, with the aim to reach high accuracy and identify plant disease. The performance of the machine learning model in a given task is calculated by a performance metric that is improved with training over time. At the end of the training period, the model can be used to predict a plant disease. For this research, data has been collected for 5 diseases that are most widespread and occur in several species of plants. Total of twenty CNN models were trained with the goal for achieving maximum accuracy by each model. Architectural configurations for each model have been designed with non-repetitive architecture. Each model is trained with a diversity of data, so as to maximize the accuracy of the result. Post training the models, each model was made to vote in decreasing order for all 5 diseases. The voting information was then accumulated and again processed to calculate the resultant disease.

The rest of the paper is organized as follows. Section 2 provides the literature review on the previous works on CNN ensemble methods. In section 3, preferential voting method is reviewed with illustration and various weight vectors are introduced. Section 4 provides the experimental results on the plaint database. Finally, section 5 concludes this work.

2 Previous Work on CNN Ensemble

CNN ensemble method with weighted voting was used in many applications, one such application is Pneumonia detection with weighted voting [5]. This paper proposes a method to detect lung opacities which can be identified as Pneumonia, on chest (CXR), by using ensemble of different classifiers with majority weighted voting ensemble method. Various attempts were done to ensemble Mask R-CNN and RetinaNet models which resulted in a higher mean accuracy precision (mAP). The suitable ratio of weights were given to each classifier which played an important role, thus increasing mAP to 0.21746.

In [13], an ensemble CNN was proposed on detection and recognition of Traffic Signs. It was developed to help the drivers in understanding what the road sign says without shifting their focus off the road. This project was done using the Belgium and German data sets. Three CNNs were trained with random initialization of weights and later aggregated to form a single ensemble model by averaging the output of each CNN.

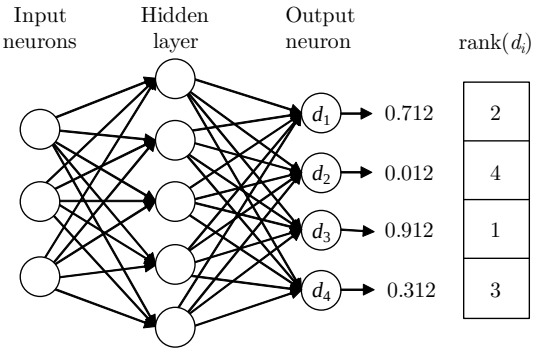
In [12], a two-stage ensemble for deep CNN was proposed. For each basic CNN model, multiple rounds of training were conducted. To increase the diversity 9 different CNN models were used. In the first stage outputs from each basic CNN were integrated using Min-Max median. This is followed by second stage by combining all outputs of each CNN model.

Another Ensemble of CNN with long-short term memory (LSTM) method was proposed by using the output of one single CNN as input to LSTM for a moment to ensemble on CIFAR-10 and CIFAR-100 dataset [3].

In [8]., the ensemble method is applied to Steganalysis. An ensemble method to combine CNN with SRM-EC model by averaging their output classification probability was proposed.

Lee et al. suggested model selecting and box voting method of two-stage detectors for the purpose of improvement in the accuracy in object detection [6]. In the proposed model object size is also considered as a selecting feature extractor. In box voting the per class Accuracy Precision (AP) was considered as weights for classifiers on the PASCAL VOC 2012 dataset, thus the results showed an im-prove in mAP.

3 Preferential Voting Methods



(a) A sample ANN and its outputs.

Classifiers

classes	c_1	c_2	c_3	c_4	c_5
d_1	2	3	4	3	1
d_2	4	1	1	4	3
d_3	1	2	2	2	2
d_4	3	4	3	1	4

(b) Sample preferential voting

Fig. 1. A sample ANN and sample preferential voting table.

Consider a sample artificial neural network with four output neurons. Suppose that there are four classes, $S = \{d_1, d_2, d_3, d_4\}$ and each output neuron

corresponds to respective class. The output is classified to the class whose output net value is the maximum. In Fig. 1 (a), the third output neuron has the highest output net value and thus it is classified as d_3. So as to utilize the preferential voting method, the rank of the classes is recorded instead of simply the highest one.

Suppose that there are ($n = 5$) different classifiers. Typical classifiers may return only the predicted class only whose output neuron value is highest. If the simple majority voting is conducted, it classifies the input instance into d_2 class as it received two votes while others received one vote each. A sample preferential voting is given in Fig. 1 (b). It ranks classes by each classifiers.

Due to flaws in the simple top choice voting system, the preferential voting system in which the voter ranks candidates in order of preference has been proposed. Amongst, the Borda method, also known as the rank method, uses the score function with certain weights. In a preferential voting, a voters ranking is an assignment of grades (e.g., "1st position", "2nd position", "3rd position") to the candidates. Requiring voters to rank all the candidates means that (1) every candidate is assigned a grade, (2) there are the same number of possible grades as the number of candidates, and (3) different candidates must be assigned different grades. A conventional Borda method uses the following weights to compute the score:

$$\text{Borda (linear: offset 0): } L_0 \qquad w_{L_0}(r) = |S| - r \tag{1}$$

$$\text{Borda (linear: offset 1): } L_1 \qquad w_{L_1}(r) = |S| - r + 1 \tag{2}$$

The class for an input instance is determined by the class with the maximum total weights. Let $r(x, d_i, c_j)$ be the rank of the ith class d_i by the jth classifier c_j on the input instance x. Then the ensemble preferential classifier can be defined as follows:

$$\text{M-classifier}(x, \{c_1, \cdots, c_n\}) = \operatorname*{argmax}_{d_i \in S} \sum_{j=i}^{n} w(r(x, d_i, c_j)) \tag{3}$$

$$= \operatorname*{argmax}_{d_i \in S} \text{score}(d_i) \tag{4}$$

$$\text{where} \quad \text{score}(d_i) = \sum_{j=i}^{n} w(r(x, d_i, c_j)) \tag{5}$$

The score for each class is defined by the sum of weights. If weight vector in eqn (1) is used in the sample voting in Fig. 1 (b), scores for each class are:

$$\begin{aligned}
\text{score}(d_1) &= 2 + 1 + 0 + 1 + 3 &&= 7 \\
\text{score}(d_2) &= 0 + 3 + 3 + 0 + 1 &&= 7 \\
\text{score}(d_3) &= 3 + 2 + 2 + 2 + 2 &&= 11 \\
\text{score}(d_4) &= 1 + 0 + 1 + 3 + 0 &&= 5
\end{aligned}$$

While the simple majority voting method classifies the input instance in Fig. 1 (b) to d_2, the Borda preferential voting method classifies it as d_3. The

Table 1. Weight vectors for various preferential voting methods.

	majority	Borda Linear		Triangular		Quadratic		Power
rank	SM	L_0	L_1	T_0	T_1	Q_0	Q_1	P_0
1	1	3	4	6	10	9	16	8
2	0	2	3	3	6	4	9	4
3	0	1	2	1	3	1	4	2
4	0	0	1	0	1	0	1	1

voting method discussed in this section can be viewed as generalizations of scoring methods, as Borda Count, i.e., different weights such as in eqn (2) can be used. While the eqn (1) produces the weight vector (3,2,1,0), eqn (2) produces the weight vector (4,3,2,1). The difference is the initial starting value, which is sometimes called *offset*.

If the weight vector is (1,0,0,0), it is the simple majority voting method since it only take the top choice only. Instead of linearly increasing function as in eqns (1) and (2), different increasing functions such as triangular number, quadratic, or power given in eqns ($6 \sim 10$) can be used to produce weight vectors.

$$\text{Triangular (offset 0): } T_0 \qquad w_{T_0}(r) = \frac{w_{L_0}(r)(w_{L_0}(r)+1)}{2} = \frac{(|S|-r)(|S|-r+1)}{2} \tag{6}$$

$$\text{Triangular (offset 1): } T_1 \qquad w_{T_1}(r) = \frac{w_{L_1}(r)(w_{L_1}(r)+1)}{2} = \frac{(|S|-r+1)(|S|-r)}{2} \tag{7}$$

$$\text{Quadratic (offset 0): } Q_0 \qquad w_{Q_0}(r) = w_{L_0}(r)^2 = (|S|-r)^2 \tag{8}$$

$$\text{Quadratic (offset 1): } Q_1 \qquad w_{Q_1}(r) = w_{L_1}(r)^2 = (|S|-r+1)^2 \tag{9}$$

$$\text{Power (offset 0): } P_0 \qquad w_{P_0}(r) = 2^{w_{L_0}(r)} = 2^{(|S|-r)} \tag{10}$$

Table 1 enumerates eight weight vectors: the simple majority (SM), Borda linear, triangular, quadratic, and power weight vectors depending on the offset. Table 2 show the scores for each class on various preferential voting methods.

Table 2. Scores for each class on various preferential voting methods.

	Classifiers					Voting methods							
classes	c_1	c_2	c_3	c_4	c_5	SM	L_0	L_1	T_0	T_1	Q_0	Q_1	P_0
d_1	2	3	4	3	1	1	7	12	11	23	15	34	17
d_2	4	1	1	4	3	2	7	12	13	25	19	38	20
d_3	1	2	2	2	2	1	11	16	18	34	25	52	24
d_4	3	4	3	1	4	1	5	10	8	18	11	26	14

4 Experiment on Plant Disease

This section describes, the entire procedure of the experiment to justify the superiority of the proposed voting method. First, the plant disease dataset is explained. The complete process of developing the convolutional neural network models and application of voting method are presented.

4.1 Plant Disease Dataset

Agricultural demands are multiplying at an alarming rate directly in proportion to the increasing population. However, the supply graph can almost be computed inversely. Unpredictable weather change and disease along with added factors cause massive yield reduction during pre-and post-harvest periods. Non-implementation of modern technology is a contributing factor to low yield. The problem of identifying plant contamination adopting methodologies has gained its attention.

The plant disease dataset contains a total of 2500 pictures of infected plants whose disease is known. The pictures were collected from google images, Plant Village dataset [9] and Citrus database [11]. There are five diseases: Anthracnose, Bacterial Spot, Black Rot, Early Blight, and Late Blight. There are nine crops: Banana, Cabbage, Grapes, Green Bell pepper, Guava, Mango, Potato, Tomato, and Yellow Bell pepper.

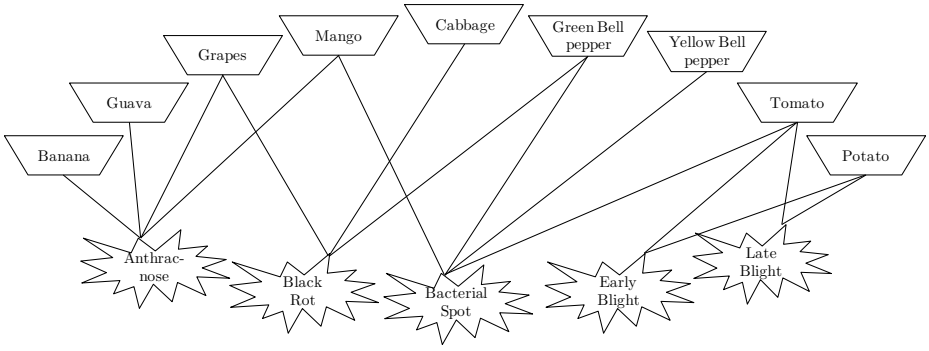


Fig. 2. Relationship between plants and diseases.

Not all diseases occur in all plants as depicted in Fig. 2. Detecting plants may help detecting diseases better. However, since the main purpose of the experiment is to compare the different voting methods' performances, the experiment here is only concerned to detect diseases only purely using multiple CNNs. Utilizing semantics and knowledge to improve the performance is beyond the scope of this paper. It can be stated that the experimental system can be utilized as a low-level sub-system to the larger disease diagnosis system.

Cropping the region of interest in the original plant image must be done as a image preprocessing. The selected region of interest is converted into 224 pixel where each pixel in an image has a value between 0 and 1.

4.2 Convolutional Neural Network Setup

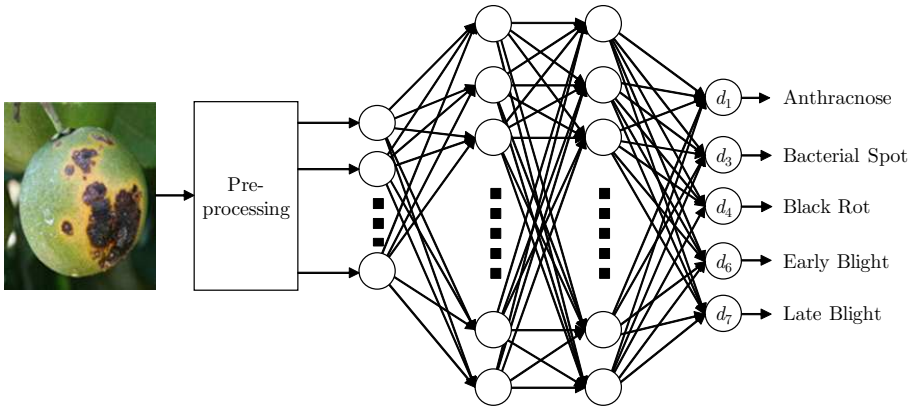


Fig. 3. CNN model for the plant diseases.

The data was divided into three parts: training data, validation data and testing data. Training and validation datasets were to be used while training of the neural network.

In order to increase the diversity, twenty different types of CNN model were defined each with a different structure. A CNN model is determined by its number of layers as well as the number of filters in each layer. The 20 CNN models with their structures are given below in Table 3. For example, the CNN c_1 has

Table 3. 20 CNN structures.

CNN	Structure	CNN	Structure
c_1	(64,128,256,512,512)	c_2	(64,128,128,256,256,256,512,512,512)
c_3	(128,256,512)	c_4	(64,256,512,1024)
c_5	(64,128,256,128)	c_6	(128,256,256,512,512,512)
c_7	(94,192,256,512)	c_8	(128,128,256)
c_9	(32,64,64,128,128)	c_{10}	(94,94,192,192,192)
c_{11}	(64,128,256)	c_{12}	(32,64,128)
c_{13}	(32,64,128,256,512)	c_{14}	(32,64,64,128,128,128)
c_{15}	(94,94,192,192,192,256,256,256)	c_{16}	(94,192,256,512,512)
c_{17}	(128,256,512,1024)	c_{18}	(64,128,256,512,1024)
c_{19}	(128,256,512,512)	c_{20}	(94,192,256,256)

five convolutional layers with 64 neurons in first layer,128 neurons in second layer, 256 neurons in third layer, 512 in fourth layer and 512 in fifth layer.

The Experiment was conducted on five classes: Late Blight, Early Blight, Bacterial Spot, Black Rot, and Anthracnose. Each class consists of 500 images per class. The entire dataset contained 2500 pictures. 300 pictures of each class were used for training all the classifiers and 50 pictures of each class were selected for validation. A total of 150 pictures of each class was used for testing the classifiers and for acquiring each classifiers vote. Hence, the total training data consisted of 1500 pictures, total validation data consisted of 250 pictures, and total testing data consisted of 750 pictures.

20 different CNN classifiers were trained with the dataset each with a different architecture as enumerated in Table 3. The accuracy of all the classifiers ranged between 65% and 80% with highest accuracy being 80% and the average of these classifiers is 76%. The simple majority voting provides considerable improvement with a performance of 82.917%. Very little improvement was recorded with Borda linear voting method with offset 0 and 1 as it showed an accuracy of 83.033%. Other voting functions like the Triangular Sequence with offset 0 and 1 did not provide much improvement. The Power Sequence voting showed the highest accuracy. These experimental results are summarized in the following Table 4. The $\max(c_x)$ being the highest accuracy amongst the 20 classifiers and $\text{ave}(c_x)$ is the average accuracy of 20 classifiers.

Table 4. Experimental Results of multiple CNNs

Classifier		Accuracy	Classifier		Accuracy
maximum	$\max(c_x)$	80.000%	average	$\text{ave}(c_x)$	76.000%
simple majority	SM	82.917%	Power	P_0 (10)	84.400%
Borda linear	L_0 (1)	83.000%	Borda linear	L_1 (2)	83.033%
Triangular	T_0 (6)	82.667%	Triangular	T_1 (7)	82.667%
Quadratic	Q_0 (8)	82.533%	Quadratic	Q_1 (9)	82.667%

5 Conclusions

Ensemble of multiple classifiers are one of the most promising methods in computer vision and machine learning. This paper proposed a better weight vector to improve the preferential voting based ensemble method that combines multiple classifiers. The plant database was used to show the experimental superiority of the proposed method.

There are many methods in computer science especially using artificial intelligence for plant disease detection and classification process, but still, research in this field is lacking. Even popular convolutional neural networks did not provide a good result. However, twenty CNNs are trained individually and combined, the good performance was observed.

The highest performance achieved using the Power Voting method is 84.40% which seems to be low. It should be noted that detecting plant diseases is often time-sensitive process. Very accurate disease diagnosis is possible by expensive and time-consuming lab tests. The proposed system is prompt and economic as it utilizes the visual information only.

There are several future works and open problems. The first one is exploring other datasets to support the superiority of the new voting method. The second one is to explore various other preferential voting method. Finally, collecting a larger plant image data is necessary.

References

1. Cha, S.-H. and An, Y. J., "Taxonomy and Nomenclature of Preferential Voting Methods," Lecture Notes in Engineering and Computer Science, *Proceedings of the World Congress on Engineering and Computer Science* (WCECS), San Francisco, USA, pp173-178, 2012.
2. Cha, S.-H. and An, Y. J., "Syntatic and Semantic Taxonomy of Preferential Voting Methods," IAENG Transactions on Engineering Technologies, Lecture Notes in Electrical Engineering, vol. 247, pp 301-315, 2014.
3. Chen, J.,Wang, Y. , Wu, Y., and Cai, C., "An Ensemble of Convolutional Neural Networks for Image Classification Based on LSTM," *Proceedings of International Conference on Green Informatics* (ICGI), Fuzhou, 2017.
4. Ho, T. K., Hull, J.J., and Srihari, S.N., "Decision Combination in Multiple Classifier Systems," IEEE Trans. J. PAMI, vol. 16, no. I, pp. 6675, 1984.
5. Ko, H., Ha, H., Cho, H., Seo K., and Lee, J., "Pneumonia Detection with Weighted Voting Ensemble of CNN Models," *Proceedings of 2nd International Conference on Artificial Intelligence and Big Data* (ICAIBD), Chengdu, China, 2019.
6. Lee, J., Lee, S. and Yang, S., "An Ensemble Method of CNN Models for Object Detection," *Proceedings of International Conference on Information and Communication Technology Convergence* (ICTC), Jeju, Korea 2018.
7. Levin J. and Nalebuff, B., "An Introduction to Vote-Counting Schemes," Journal of Economic Perspectives, vol. 9, no. I, pp. 326, 1995.
8. Liu, K., Yang, J., and Kang, X., "Ensemble of CNN and rich model for steganalysis," *Proceedings of International Conference on Systems, Signals and Image Processing* (IWSSIP), Poznan, 2017.
9. Mohanty, S. P., "Dataset of diseased plant leaf images and corresponding labels," https://github.com/spMohanty/PlantVillage-Dataset, 2018.
10. Park, E., Han, X., Berg, T. L.. and Berg, A. C., "Combining multiple sources of knowledge in deep CNNs for action recognition," *Proceedings of IEEE Winter Conference on Applications of Computer Vision* (WACV), Lake Placid, NY, 2016.
11. Rauf, H. T., Saleem, B. A., Lali, M. I. U., Khan, Sharif, M. A., M., and Bukhari, S. A. C., "A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning," *Data in Brief*, vol. 26, Article104340 , 2019.
12. Uddamvathanak, R., "Two-Stage Ensemble of Deep Convolutional Neural Networks for Object Recognition," *Proceedings of International Conference on Intelligent Rail Transportation* (ICIRT), Singapore, 2018.
13. Vennelakanti, A. , Shreya, S. , Rajendran, R. , Sarkar, D. , Muddegowda D. , and Hanagal P., "Traffic Sign Detection and Recognition using a CNN Ensemble,"

Proceedings of IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2019.

14. Yamashita, R., Nishio, M., Do, R. K. G., and Togashi, K., "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 611, no. 1869-4101, p. 9, 2018.

Multiple Imputation with Denoising Autoencoder using Metamorphic Truth and Imputation Feedback

Haw-minn Lu, Giancarlo Perrone, and José Unpingco

Gary and Mary West Health Institute, San Diego, CA 92037, USA
{hlu,gperrone,jhunpingco}@westhealth.org

Abstract. Although data may be abundant, complete data is less so, due to missing columns or rows. This missingness undermines the performance of downstream data products that either omit incomplete cases or create derived completed data for subsequent processing. Appropriately managing missing data is required in order to fully exploit and correctly use data. We propose a Multiple Imputation model using Denoising Autoencoders to learn the internal representation of data. Furthermore, we use the novel mechanisms of Metamorphic Truth and Imputation Feedback to maintain statistical integrity of attributes and eliminate bias in the learning process. Our approach explores the effects of imputation on various missingness mechanisms and patterns of missing data, outperforming other methods in many standard test cases.

Keywords: unsupervised learning, neural networks, deep learning, imputation, missing data, autoencoders

1 Introduction

Missing data is problematic, ubiquitous, and lacks a universal solution. Most often, incomplete observations are simply omitted, thus shrinking the dataset, along with any potential gains from that data. One common approach is to simply replace the missing values with the mean/median of the observed values in the same partition. Other approaches use a machine learning model to preprocess and fill-in the missing observations. While these methods superficially resolve the missing data, they may also lead to downstream bias and unexpected errors.

Why data is missing is a further difficult problem, requiring domain knowledge and understanding of the data collection process. Did a corrupted input lead to an empty response in an online survey? Or was that field intentionally skipped? Understanding the mechanism of missing data drives the effectiveness of the imputation method [7]. Imputing missing observations by the mean/median fails to account for the uncertainty of missing values during the imputation procedure. To circumvent this, the process of imputing may be repeated, producing multiple imputed values that are scholastically different. This

is known as *Multiple Imputation* [9], which involves repeatedly performing imputation inferences on any missing data, and then combining the results to analyze imputed results incorporating any previous uncertainty for missing data. This procedure recognizes the uncertainty of predictions by introducing variability [10] into imputed values.

In section 2, we described the terminology and formal concepts relating to missingness as well as the previous use of autoencoders in multiple imputation. In section 3, we describe the architecture of the underlying deep neural network used. In section 4, we describe the training method. A catalog of data used is described in section 5. The experiment is detailed in section 6. The results are presented in section 7. Section 8 discusses the implications of the results.

2 Background and Related Work

2.1 Missingness Models

Consistent with recent literature, we distinguish between missingness *mechanisms* and missingness *patterns*. Introduced by Rubin, [7] missingness mechanisms refers to one of three situations: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Additionally, we adopt the table nomenclature for a dataset where *rows* are synonymous with observations, *columns* with attributes and *cells* with attributes of an observation. If a particular column cannot have any missing rows (i.e., not subject to missing data), that column is *permanent*. Otherwise, it is referred to as *vulnerable*.

- MCAR – A row has one or more missing cells independent of the values of the cells in that row.
- MAR – Unlike MCAR, a missing cell is a probabilistic function of the permanent columns. For example, the local school district surveys families regarding household income, while knowing which families participate in the school free lunch program. Families who participate in the program may be less likely to respond to the household income question. Thus, the column indicating participation in the free lunch program is *permanent* but the household income data may be missing for a particular row based on program participation.
- MNAR – All cases other than MCAR and MAR. This implies that the missingness is a function of variables in all columns, permanent or not. Thus, a cell could be eliminated probabilistically based upon its own value, which means that it does not appear in the resulting dataset. For example, an overweight respondent might be less likely to report his/her body weight so the resulting data set may not contain a value for body weight for some rows.

We refer to missingness *patterns* as random and uniform. A *uniform* missingness pattern means that all vulnerable columns in a particular row are missing. A *random* pattern means that some elements in the vulnerable columns are missing probabilistically. In summary, the missingness mechanism decides whether a particular row has missing data and the missingness pattern indicates which cells in a given row are subject to missingness.

2.2 Autoencoders

An autoencoder in the context of artificial neural networks and in particular deep neural networks, is a neural network that is trained to copy the input as the output (i.e, approximate the identity function). Rumelhart, *et. al.*[8] showed that they can be used to learn a hidden internal representation of the data. Denoising Autoencoders (DAE)[12] are designed to remove noise from data. They are characterized by high dimension in the hidden layers and with stochastic corruption (dropout) at the input. DAEs are employed by Gondara[3] to perform multiple imputation. The research presented here is based on the deep neural network employed by Gondara and is explained in greater detail below. There are additional methods employing autoencoders to attack the problem of imputation[11].

3 Architecture

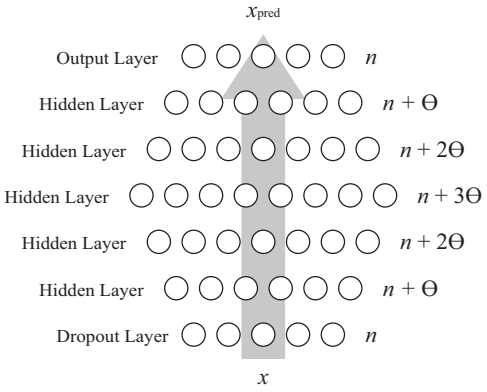


Fig. 1. Denoising Autoencoder

Our autoencoder architecture is essentially identical to that used by Gondara. The main difference lies in the training method described in the next section. Our DAE architecture employs most of the same hyperparameters as Gondara. As shown in Figure 1, the first layer implements a dropout (stochastic corruption) of 50%, there are 5 additional hidden layers with the layers first increasing in width by the hyperparameter Θ from the previous layer then decreasing by the same Θ in the final layers, where $\Theta = 7$. All of these layers, except for the output layer, use a tanh activation function. The depth of the network, the value of Θ, the activation function and degree of dropout are as recommended by Gondara. Additionally, the DAE architecture is implemented using Tensorflow and using the Adam optimizer[5].

Other factors which can be treated as hyperparameters include the initial treatment of missing data and the proportion of data used in training. Because training neural networks requires all values to be present, missing values must be imputed with some value. A more detailed discussion of initial imputation methods is given in the next section. Additionally, the proportion of data used in training is more appropriately discussed in the next section.

4 Method

The method of training the DAE is significantly different than that of training a DAE or that of a typical deep neural network. First, we discuss the motivation behind our training method and then the method of metamorphic truth and feedback. Finally, we discuss the preparing the training data.

4.1 Motivation

Deep neural networks are not designed to accommodate missing data. Therefore, imputation techniques using deep neural networks perform some type of initial imputation to fill in missing data in a training set prior to training. For example, Gondara initially imputes the data using mean-imputation for the numerical columns on an architecture identical to ours. However, the initial imputation is extremely important for downstream performance of the DAE. For example, Table 1 shows imputing the Boston Housing Data using the DAE under MAR mechanism with random pattern where initial imputation is performed five times. The mean over these imputations is shown with the corresponding maximum value over those imputations in parenthesis. The Max Imputed column uses the maximum value of the respective columns for the initial imputation, intuitively a poor choice. The Perfect Guess column fills in the true value of the missing cell for the initial imputation, an obviously unrealistic best case initial imputation. This table shows that maximum initial imputation has the worst RMSE_{sum}, which is approximately twice that of a mean-value initial imputation. Note that the Perfect Guess initial imputation is half that of the mean imputed initial imputation. This table shows that the initial imputation severely affects downstream performance of the DAE.

In this example as well as throughout our experiments RMSE_{sum} is used to measure the quality of a given imputation. The RMSE_{sum} is defined by equation 1.

$$\text{RMSE}_{\text{sum}} = \sum_{j=1}^{N_{\text{row}}} \sqrt{\mathbb{E}\left(\sum_{i=1}^{N_{\text{col}}} (\tilde{t}_i - t_i)^2\right)}, \tag{1}$$

where N_{row} are the number of rows, N_{col} are the number of columns, $\tilde{t}_i$ is an imputed cell, and t_i is the corresponding cell from the original uncorrected row and $\mathbb{E}$ is the expected value or mean. Note that only imputed cells appear in this equation.

There are two factors which contribute to this sensitivity to initial imputation. First, the initial imputed value is fed back to the neural network as the ground truth during training. This essentially tells the DAE that the *correct* answer is the *initial* imputed value, which biases the DAE towards the initial imputed value, and potentially away from a better value. As a practical matter, for many datasets, the mean is a good choice for imputation when considering *mean* squared error. Because of this, if the mean is close to the good imputed value the feedback to the DAE will have less impact.

The second factor is that as the DAE produces better imputed values, the input and output diverge, which is *contrary* to the learning objective for the DAE, which is to reproduce the input as the output, but without the noise.

Table 1. Effect of Initial Imputation on DAE Imputation

	Mean Imputed	Max Imputed	Perfect Guess
RMSE_{sum}	7.43 (7.59)	16.72 (16.92)	3.16 (3.32)

4.2 Metamorphic Truth and Feedback

Since an initial imputation must be performed to use the DAE, we use mean imputation. However, because our method mitigates the two factors contributing to initial imputation sensitivity mentioned above, the choice of the initial imputation is not as important. To mitigate the first factor, *metamorphic truth* is applied and to mitigate the second factor we feedback interim imputed values as the imputation for inputs to subsequent training epochs. In short, metamorphic truth decouples the DAE's output from the initial imputation and the feedback decouples the DAE's input from the initial imputation.

Figure 2a shows the typical training used traditionally with DAE in imputation. Given a row $\mathbf{x}$, missing values are imputed as $\tilde{\mathbf{x}}_0$. The initial imputed value $\tilde{\mathbf{x}}_0$ is fed into the DAE which produces a prediction $\mathbf{x}_{\text{pred}}$ (we keep the notation of $\mathbf{x}$ rather than $\mathbf{y}$ as this is an autoencoder). A mean squared error (MSE) loss is computed by equation 2

$$L = ||\tilde{\mathbf{x}}_0 - \mathbf{x}_{\text{pred}}||^2 \tag{2}$$

and then fed back into the DAE to be used to adjust weights by a proscribed optimization algorithm. Figure 2b shows the modified learning process used to address the two aforementioned issues. To address the first concern mentioned above, we employ a technique called *metamorphic truth*. Metamorphic truth is a technique used to change the truth reported to the optimizer based on the prediction, but having the neural network treat the *truth metamorph* $\bar{x}$ as a constant. This distinguishes the technique from simply being a more complicated loss function.

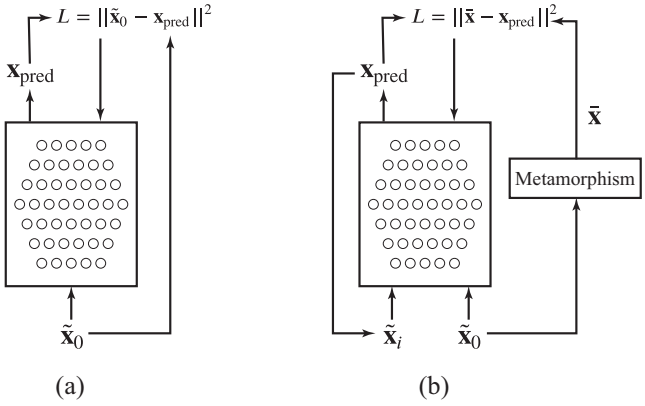


Fig. 2. Denoising Autoencoder Learning

For imputation we use the following metamorphism, $\mathcal{M}$:

$$\mathcal{M}(\mathbf{x}, \mathbf{x}_{\text{pred}})_i = \begin{cases} x_{\text{pred}_i} & \text{If } x_i \text{ is a missing attribute} \\ x_i & \text{Otherwise} \end{cases} \tag{3}$$

For the i^{th} cell in a row, the truth metamorph is therefore expressed as

$$\bar{x} = \mathcal{M}(\tilde{\mathbf{x}}_0, \mathbf{x}_{\text{pred}}). \tag{4}$$

and the loss function can now be expressed as

$$L = ||\bar{\mathbf{x}} - \mathbf{x}_{\text{pred}}||^2. \tag{5}$$

We address the second aforementioned issue (divergence between input and output) by using the predicted values to impute subsequent training of the DAE. The new learning algorithm is given by Algorithm 1 which has two phases, (1) a priming phase and (2) a fine-tuning phase. In the priming phase, the DAE is trained for N_{prime} epochs on the initial imputed values (we initially impute with the mean). The purpose of this phase is to coarsely train the DAE, in principle 10-20 epochs is found to be sufficient for the datasets considered here. In the fine-tuning steps, the output of the DAE is used to compute a new imputation, and that imputed dataset is then used to train the DAE for N_{step} epochs. The new imputation only takes the predicted output for the *missing* attributes and leaves *observed* attributes alone. This process is repeated for the desired number of steps. In our experiments, $N_{\text{step}} = 2$ though $N_{\text{step}} = 1$ can be used.

4.3 Training Methodology

In a typical deep neural network training process, part of a dataset is held back for validation purposes. For example, Gondara separates datasets samples into

Algorithm 1 Imputation

for Number of imputations required **do**
  Set initial imputation $\tilde{\mathbf{x}}_0$
  Train DAE with $\tilde{\mathbf{x}}_0$ for N_{prime} epochs
  Run DAE on $\tilde{\mathbf{x}}_0$ to produce $\mathbf{x}_{\text{pred}}$
  for $i = 1 \ldots$ Number of steps required **do**
    Impute $\tilde{\mathbf{x}}_i$ using $\mathbf{x}_{\text{pred}}$
    Train DAE with $\tilde{\mathbf{x}}_i$ for N_{step} epochs
    Run DAE on $\tilde{\mathbf{x}}_i$ to produce $\mathbf{x}_{\text{pred}}$
  end for
  take last $\tilde{\mathbf{x}}_i$ as the imputed dataset
end for

approximately 70% training set and 30% test set. The main purpose of a validation data is to gauge when a neural network is starting to overfit. In short, the validation set is used to indicate when to stop training. In an *open learning* system, overfitting can be detrimental because the neural network is will not adapt to future unknown inputs. Hence validation data is used to measure "model accuracy," the accuracy applied to the entire open ended problem. However, DAEs used for imputation are *closed learning* systems, that is once trained no future unknown inputs need to be accounted for. Hence, the detrimental effect of potential overfitting is not significant in a closed learning system and the model accuracy can be measured only on the training set since the training set represents the entire data set both present and future.

Beyond the use of a test set to detect overfitting in an open learning system, a test set is also used to evaluate the accuracy of a given deep neural network model. For the experiments presented here, true accuracy of the DAE is reflected in how well the imputed values compare to the original values that are removed as part of the experiment with measures such as RMS_{SUM} and covariance drift as defined below. Therefore, there is no need for a test set to be held in reserve to evaluate model accuracy as such model accuracy may not reflect the actual accuracy of the imputation.

Since there is no longer a compelling reason to reserve some samples for a test set, we elected to use **all samples** for training. As a test set would not influence DAE training, omission of samples reserved in a test set would risk eliminating potential covariance relationships or other joint statistical relationships present in the omitted samples.

5 Data

Six datasets are used for the evaluation of the models are taken from [2] and [4][1]. Five of the six are used by Gondara. Table 2 shows the name of the datasets

[1] obtained at http://lib.stat.cmu.edu/datasets/boston

with their abbreviations along with the number of columns and rows[2] in the dataset. For the same reasons as in the previously cited work, the motivation for the database selection is to demonstrate the effectiveness of the imputation model even under low-dimensionality or low sample size datasets.

We adopt the convention set forth by Li[6] for inducing missingness in the datasets. For each dataset, columns are designated permanent (i.e., not subject to missingness) and vulnerable (i.e., subject to missingness). The specific designation of attributes for each category was arbitrary except that *only* numeric and *not* categorical columns were classified as vulnerable, in order to keep the experiment simple. However, categorical columns are retained as permanent columns so that they may potentially serve as covariates for imputation. Proper consideration of categorical attributes would entail exploring proper ways to encode those categories.

Table 2. Datasets

Name	Rows	Columns
Boston Housing (BH)	506	14
Glass (GL)	214	10
Ionosphere (IS)	351	35
Breast Cancer (BC)	683	11
Sonar (SN)	208	61
Wine (WN)	4898	11

6 Experiments

6.1 Inducing Missingness

Missingness is induced using the three mechanisms described above, MCAR, MAR and MNAR. In addition we apply a missingness pattern of either random or uniform. Our method of inducing the missingness involves two tuning probabilities p_m and p_p, the former is used in the mechanism and the latter, the pattern. These are probabilities that can be adjusted to obtain the desired level of missingness and do not necessarily reflect on the probability an attribute is missing.

For MCAR missingness, a row has missing values with probability p_m. Clearly this is purely random and missingness is not a function of the values in the cells. Our construction of MAR and MNAR missingness is algorithmically the same, two random columns (i, j) are selected for the entire dataset. If for a given row $\mathbf{x}$ has $x_i > \mu_i$ and $x_j > \mu_j$ where μ_i and μ_j are the means of those two

[2] It should be noted that the BC dataset had 699 rows which included 16 rows with missing cells. Those observations were dropped from the dataset.

columns, then the observation has missing values with probability p_m. For MAR, the two selected columns are drawn from only the permanent columns and for MNAR they are drawn from the vulnerable columns. In the case of the uniform missingness pattern, if a row is induced by the mechanism to have missing data then the *entire* row of the across all vulnerable columns is removed; otherwise, for the random missingness pattern, the cells along that row are removed by the missingness mechanism with with probability p_p.

For each dataset, a case of MCAR uniform, MCAR random, MAR uniform, MAR random, MNAR uniform and MNAR random are induced and saved so that the four imputation methods mentioned below can be applied to the same corrupted datasets. Generally our target for all cases is to induce a missing proportion between 14% and 20%[3] achieved by adjusting the tuning probabilities. The actual percentage of cells with missing values are given in Table 3.

Table 3. Percentages of Cells Missing

	Random			Uniform		
Dataset	MCAR	MAR	MNAR	MCAR	MAR	MNAR
BH	15.8%	17.8%	15.2%	16.1%	17.0%	15.5%
GL	14.4%	14.0%	14.2%	16.1%	14.2%	14.1%
IS	20.4%	14.6%	10.0%	14.2%	12.1%	10.6%
BC	16.4%	19.4%	18.1%	16.7%	15.1%	16.1%
SN	16.0%	15.9%	10.2%	16.7%	14.7%	12.1%
WN	16.1%	14.0%	11.8%	14.3%	10.2%	10.7%

6.2 Comparison to Other Methods

Comparative experiments are conducted comparing four imputation models. First is mean imputation where the mean of an attribute is substituted for the missing values. Second is Multivariate Imputation by Chained Equations (MICE) [1], which is considered the state-of-the-art multiple imputation model. Third is the DAE presented by Gondara, which our model relies heavily upon. Fourth is the DAE with Metamorphic Truth and Imputation Feedback (DAE MT) presently described above.

All measures applied to the quality of the imputations are computed on datasets that are normalized. For all imputation methods except MICE (MICE does not need it), data is normalized prior to the application of the model. However, in the case of MICE, the imputed output data is normalized using the same scaling factors used to normalize the other methods for fair comparison. Normalization is performed on each cell by subtracting the mean of that column and dividing by the standard deviation. The MICE program was run using

[3] However in some cases, reaching that percentage proved challenging so the proportion of missing data for those cases we lowered the threshold to 10%

Table 4. Comparison of imputation techniques (RMS_{SUM})

	D.set	Random				Uniform			
		DAE MT	DAE	MICE	Mean	DAE MT	DAE	MICE	Mean
MCAR	BH	**6.7** (6.8)	8.0 (8.1)	8.4 (8.8)	9.4	**6.5** (6.6)	7.8 (7.9)	9.9 (10.3)	8.4
	GL	**4.3** (4.4)	4.9 (5.1)	5.8 (6.8)	5.7	**4.4** (4.5)	5.0 (5.1)	7.3 (8.5)	5.6
	IS	**17.7** (18.0)	20.7 (20.9)	23.1 (23.6)	23.3	**21.6** (21.7)	23.9 (24.1)	27.4 (28.4)	25.9
	BC	**4.4** (4.5)	5.6 (5.6)	6.1 (6.3)	6.7	**5.3** (5.4)	7.1 (7.2)	6.6 (6.9)	7.7
	SN	**28.6** (28.9)	34.2 (34.3)	37.3 (38.5)	39.5	**37.4** (37.5)	39.1 (39.2)	49.9 (50.6)	39.7
	WN	**5.5** (5.6)	6.2 (6.3)	8.0 (8.1)	6.9	**6.2** (6.3)	6.8 (6.8)	8.8 (9.1)	7.1
MAR	BH	**5.4** (5.5)	7.4 (7.5)	7.9 (8.4)	8.4	**5.8** (5.9)	8.3 (8.4)	10.0 (10.3)	8.8
	GL	**2.8** (2.9)	3.5 (3.6)	6.2 (7.6)	4.1	**7.4** (7.4)	7.8 (7.8)	10.2 (10.7)	8.1
	IS	**19.0** (19.2)	19.9 (20.0)	25.5 (26.2)	21.3	**17.2** (17.3)	19.4 (19.5)	24.1 (24.7)	21.4
	BC	**2.1** (2.4)	4.4 (4.6)	2.4 (2.8)	5.4	**3.2** (3.3)	4.7 (4.8)	4.3 (4.5)	5.1
	SN	**35.5** (35.7)	39.9 (40.2)	50.8 (52.7)	41.5	**36.1** (36.3)	38.9 (39.2)	50.3 (51.6)	39.4
	WN	**5.8** (5.8)	6.3 (6.3)	8.5 (8.5)	6.7	**5.9** (5.9)	6.3 (6.3)	8.6 (8.7)	6.6
MNAR	BH	**6.6** (6.7)	8.4 (8.5)	8.8 (8.9)	9.4	**9.4** (9.6)	12.9 (13.1)	12.2 (12.4)	13.6
	GL	**3.1** (3.2)	3.9 (3.9)	5.6 (6.3)	4.4	**4.2** (4.3)	5.1 (5.2)	6.6 (7.2)	5.6
	IS	**17.6** (17.7)	20.7 (21.0)	23.0 (23.6)	23.3	**21.5** (21.8)	25.7 (26.1)	27.4 (27.8)	28.2
	BC	**2.2** (2.3)	4.5 (4.6)	2.6 (2.8)	5.5	**2.7** (2.9)	4.8 (4.9)	3.0 (3.3)	5.3
	SN	**40.1** (40.4)	46.1 (46.5)	53.4 (55.1)	49.4	**34.0** (34.5)	36.0 (36.2)	48.2 (49.7)	36.7
	WN	**5.6** (5.6)	6.1 (6.1)	8.5 (8.5)	6.5	**6.9** (7.0)	7.7 (7.8)	9.4 (9.5)	8.1

default settings with 5 imputations. The DAE was trained for 500 epochs. The DAE MT was primed for 10 epochs (N_{prime}) and 245 steps of 2 epochs (N_{step}) each, totalling the same 500 epochs as the DAE. Both DAE MT and DAE were run 5 times for 5 imputations.

7 Results

Table 4 shows the results from running the various models on all the datasets. For each multiple imputation model, the mean and the maximum error (RMS_{SUM}) across the imputations are shown with the maximum in parenthesis and the lowest value shown in bold. Since mean imputation is a single imputation, a single error is reported. From the table, DAE MT outperforms both standard DAE, MICE and mean imputation in all cases studied. In many applications such as machine learning applications such as classification, minimizing RMS_{SUM} is very important, but for more traditional statistical analytics, the relationship between values in different columns is most important.

As a rough measurement to the how well these relationships are preserved, we defined a *covariance drift*, which is basically the RMS average of the difference between the respective covariances of the original data and that of the data post-imputation. Mathematically, if the elements of the covariance matrix of the uncorrupted data set is $\sigma_{i,j}$ and the elements of the covariance matrix of the imputed data set is $\tilde{\sigma}_{i,j}$ then the drift is given by equation 6.

$$\text{drift}_{\text{cov}} = \frac{1}{N(N-1)} \sqrt{\sum_{i \neq j} (\sigma_{i,j} - \tilde{\sigma}_{i,j})^2} \tag{6}$$

In the interest of space, Table 5 shows only the covariance drift for the MNAR missingness case for all the datasets. Once again, the mean and the maximum values (in parenthesis) of the covariance drift are shown with the lowest drift values in bold. The results are multiplied by 10 to be displayed in the table for easier comparison. With regard to the covariance drift, MICE generally outperforms both DAE models, however DAE MT outperforms significantly standard DAE in all cases and in some cases exceeds the performance of MICE. Hence, it is clear that the bias asserted by the initial imputation has a strong impact on the covariance which is significantly mitigated by DAE MT. Unsurprisingly, MICE does best in minimizing covariance drift in general as MICE strives to preserve these relationships whereas neural networks aim to minimize mean squared errors.

Table 5. Covariance Drift under MNAR Missingness (×10)

	Random				Uniform			
Dataset	DAE MT	DAE	MICE	Mean	DAE MT	DAE	MICE	Mean
BH	0.6 (0.6)	1.1 (1.2)	**0.4** (0.5)	1.3	1.3 (1.5)	2.6 (2.6)	**1.2** (1.3)	2.7
GL	**0.4** (0.4)	0.6 (0.6)	0.5 (0.6)	0.7	**0.7** (0.7)	0.9 (1.0)	0.8 (0.9)	1.0
IS	0.6 (0.6)	0.7 (0.7)	**0.5** (0.6)	0.8	0.8 (0.8)	1.1 (1.1)	**0.6** (0.7)	1.2
BC	0.4 (0.5)	1.1 (1.2)	**0.2** (0.2)	1.4	0.4 (0.5)	0.9 (0.9)	**0.1** (0.2)	1.0
SN	**0.5** (0.5)	0.7 (0.7)	0.6 (0.7)	0.8	**0.4** (0.4)	0.5 (0.5)	0.5 (0.5)	0.5
WN	**0.4** (0.4)	0.5 (0.5)	0.6 (0.6)	0.6	0.5 (0.6)	0.7 (0.7)	**0.3** (0.3)	0.8

8 Conclusion

For machine learning applications where minimizing RMS is a key factor, our DAE with metamorphic truth and imputation feedback outperforms both MICE and Gondara's DAE imputation model. For removing bias from covariance relationships, our DAE surpasses Gondara's DAE and approaches and even exceeds MICE performance. Additionally, the technique of metamorphic truth can be used to shape the way DAE or other neural networks learn. For example it may be possible to develop a metamorphism to guide the imputation to have better statistical characteristics, possibly by incorporating a regression or other MICE strategy into the metamorphism. Beyond the case of imputation, metamorphic truth could have wider applications in areas such as classification, that have yet to be explored.

References

1. van Buuren, S., Groothuis-Oudshoorn, K.: mice: Multivariate imputation by chained equations in r. Journal of Statistical Software, Articles **45**(3), 1–67 (2011). https://doi.org/10.18637/jss.v045.i03, https://www.jstatsoft.org/v045/i03
2. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
3. Gondara, L., Wang, K.: Mida: Multiple imputation using denoising autoencoders. In: Phung, D., Tseng, V.S., Webb, G.I., Ho, B., Ganji, M., Rashidi, L. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 260–272. Springer International Publishing, Cham (2018)
4. Harrison, D., Rubinfeld, D.: Hedonic Housing Prices and the Demand for Clean Air. Journal of Environmental Economics and Management **5**, 81–102 (1978)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
6. Li, L., Shen, C., Li, X., Robins, J.M.: On weighting approaches for missing data. Statistical methods in medical research **22**(1), 14–30 (Feb 2013). https://doi.org/10.1177/0962280211403597, https://www.ncbi.nlm.nih.gov/pubmed/21705435, 21705435[pmid]
7. Rubin, D.D.: Inference and missing data. In: Biometrika. pp. 581–592. Oxford University Press (1976)
8. Rumelhart, David E., H.G.E., Williams, R.J.: Learning internal representations by error propagation. Institute for Cognitive Science Report ICS-8506, University of Califorina, San Diego, La Jolla, CA (September 1985)
9. Schafer, J.L.: Multiple imputation: a primer. Statistical Methods in Medical Research **8**(1), 3–15 (1999). https://doi.org/10.1177/096228029900800102, https://doi.org/10.1177/096228029900800102, pMID: 10347857
10. Sterne, J., White, I., Carlin, J., Spratt, M., Royston, P., Kenward, M., Wood, A., Carpenter, J.: Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls. british medical journal. BMJ (Clinical research ed.) **338**, b2393 (02 2009). https://doi.org/10.1136/bmj.b2393
11. Tran, L., Liu, X., Zhou, J., Jin, R.: Missing modalities imputation via cascaded residual autoencoder. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4971–4980. IEEE (July 2017). https://doi.org/10.1109/CVPR.2017.528
12. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. p. 1096–1103. ICML '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1390156.1390294, https://doi.org/10.1145/1390156.1390294

Deep 3D Face Recognition using 3D Data Augmentation and Transfer Learning

Lyndon N. Smith[0000-0001-5821-0586] Ning Huang, Mark F. Hansen and Melvyn L. Smith

Centre for Machine Vision, Bristol Robotics Laboratory, Department of Engineering Design and Mathematics, University of the West of England, Bristol, BS16 1QY, UK
lyndon.smith@uwe.ac.uk

Abstract. Deep convolutional neural networks (DCNNs) have achieved human-comparable performance on challenging 2D face databases, outperforming all previous shallow methods. However, current 3D face recognition research still focuses on non-deep-learning methods due to the lack of large-scale 3D face databases. To address this, this paper proposes new 3D data augmentation methods: pose-based and channel-based augmentation. Experiments on three databases show that deep convolutional neural networks can be used effectively for 3D face recognition. Using a pose-augmented training set, an eight-layer convolutional neural network achieved 100%, 99% and 99% of validation accuracy on Photoface-10, FRGC-10 and Photoface-50 databases respectively. Also, successful channel-based augmentation, with five more artificial sessions per a three-channel image, showed that feature extraction in DCNNs is channel-invariant. It was found that transfer learning from a pre-trained deep neural network (e.g. VGG19) works for 3D face recognition, by fine-tuning the last few fully connected layers using 3D facial scans. The performance of a bespoke DCNN was compared to a fine-tuned pre-trained DCNN. The bespoke DCNN could achieve competitive performance if enough training data were provided; however, transfer learning from a pre-trained model provides an advantage in training time, being at least 3 times faster.

Keywords: 3D data augmentation, deep convolutional neural networks, transfer learning, face recognition.

1 Introduction

In contrast to 2D face images, 3D facial models are insensitive to illumination and natural makeup [1]; and the 3D surface information makes spoofing much harder (2D face recognition system can be tricked easily by substitutions of face photographs). Despite these advantages, 3D (unlike 2D) face recognition employing deep learning, has not been the subject of intensive research. Consequently, most 3D face recognition algorithms still use non-deep-learning methods to extract distinct features from 3D facial models [2]. This may be due to the large amount of data traditionally needed for training convolutional neural networks and the general lack of large-scale 3D face databases. The latter may be due to the time and expense involved in capturing 3D face data,

compared to the ease with which 2D data can be sourced directly from the Internet. For example, researchers from the University of Oxford assembled a large-scale 2D face database (2.6M images, over 2.6K people) using Google Image Search [3]; while FaceNet [4], one of the most famous deep 2D face recognition systems, employs 260M facial sessions of 8M subjects to train its networks. In contrast to this, the Photoface database [5], one of the largest publicly available 3D face databases, contains 'only' 3187 sessions of 453 subjects. Previous research showed that DCNNs exhibit superior performance to previous face recognition methods, and that the performance of the DCNN improves as the amount of training data increases [6]. Previous 2D work has shown that data augmentation for generating artificial face sessions assists with face recognition [7]. The challenge now is to employ augmentation to 3D face data, in order to benefit from the additional reliability and spoof-proof capabilities that DCNN based 3D face recognition can offer. The work described here endeavors to do this by applying data augmentation and transfer learning to selected 3D face databases. In Section 2 we describe the methodology, while Section 3 covers the experiments and results obtained. Finally, in Section 4 we summarize the findings of our work.

2 Methodology

Facial data from the Photoface database [5] and the FRGC database [8] were employed. Photoface, consisting of a total of 3187 sessions of 453 subjects, contains 2.5D human facial data captured using photometric stereo in an unsupervised corridor. Hence each subject was captured with natural poses and expressions. However, since it was an unsupervised data collection, the session distributions among subjects were highly unbalanced where some individuals had more than 30 scans while most of the subjects only had one. In Photoface, surface normal vector components (NX, NY, NZ) and albedo estimations were calculated and stored. A representation of facial surface normals is provided in Fig. 1, where the Z-component of the facial surface normal (NZ) is not represented since it is redundant for face recognition (being present to make the magnitude of the components sum to one) [9].

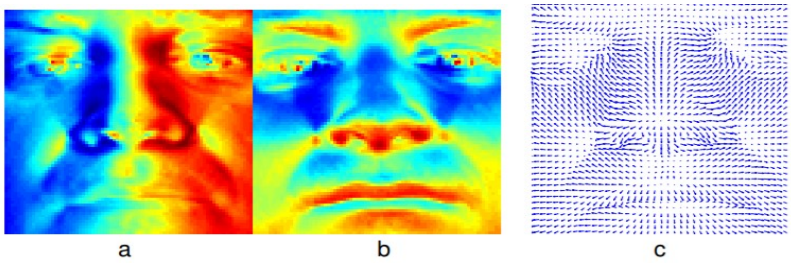


Fig. 1. Representations of facial surface normal. (a) Surface normal, X-components. (b) Surface normal, Y-components (c) Surface normal, needle representation – also known as the 'bump map' [5].

For 2D deep face recognition, color images with RGB channels were the main input of a DCNN. Inspired by this, to feed 3D surface normal data into a convolutional neural network, a method was employed that replaced the three RGB channels of a 2D color image by the facial surface normals NX and NY, and the albedo. The albedo, which is the fraction of light reflected by the surface at the point concerned, was used instead of NZ.

2.1 Data augmentation

The major challenge in deep 3D face recognition is how to train a DCNN without overfitting due to the small amount of data available compared with 2D datasets. Since existing 3D face databases are insufficient for deep neural network training, a data augmentation technology is proposed to generate artificial face sessions that enlarges the database by rotating the 3D face. However, it is not possible to do this by simply rotating the bump map – since this would not preserve the morphological integrity and geodesic relationships of the face. Therefore, it was necessary to convert each bump map into a 3D point cloud that could then be rotated. The data augmentation algorithm developed can generate more than 100 sessions at various viewpoints using one original 3D face, and hence potentially addresses the pose variation issue in face recognition. This algorithm exhibits good robustness that can be used on both the FRGC v2.0 and Photoface databases. The procedure to generate an artificial range image is as follows:

1. Extend the original bump map into 3D space, forming a point cloud model.
2. To rotate the point cloud, the coordinate of each facial point was multiplied by the rotation matrix R to achieve yaw, pitch, and row rotations, as shown in equations 1 to 4:

$$Rx(\theta 1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta 1) & -sin(\theta 1) \\ 0 & sin(\theta 1) & cos(\theta 1) \end{pmatrix} \quad (1)$$

$$Ry(\theta 2) = \begin{pmatrix} cos(\theta 2) & 0 & sin(\theta 2) \\ 0 & 1 & 0 \\ -sin(\theta 2) & 0 & cos(\theta 2) \end{pmatrix} \quad (2)$$

$$Rz(\theta 3) = \begin{pmatrix} cos(\theta 3) & -sin(\theta 3) & 0 \\ sin(\theta 3) & cos(\theta 3) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

$$R = Rx(\theta 1) * Ry(\theta 2) * Rz(\theta 3) \quad (4)$$

3. Build a triangulation model for the point cloud using X, Y and Z coordinates and interpolate the corresponding depth value (Z) for each integer coordinate on the X and Y axis (where the X and Y coordinates of points of the rotated point cloud model may not have a value that cannot be projected onto 2D image space).

4. Project the points in the cloud model, with integer X and Y coordinates, onto 2D image space to form the rotated range image.

5. Calculate 3D surface normal data (NX, NY, NZ) for the rotated 3D face. As previously stated, the Z-component of the facial surface normal was redundant for face recognition and was replaced by the facial albedo image. To construct rotated 2.5D artificial face sessions, the facial albedo needed to rotate identically to the cloud point data. Consequently, a novel method for rotating the 2D albedo image was employed. This was based on an assumption that that the pixel intensity of the 2D albedo image was continuous in 3D space so that triangulation could be used to interpolate pixel intensity. The proposed method can generate rotated face sessions with excellent quality as long as the range image rotation is less than roughly 45 degrees. Fig. 2 illustrates a series of augmented 3D images generated from the Photoface database using the method described. The 3D data are arranged in the sequence of [NX, albedo, NY].

Fig. 2. 3D data augmentation examples. The central image shows the original data - the other eight comprise augmented data generated using the proposed augmentation method; all rotations are by 20 degrees.

80 sessions belonging to 10 subjects (8 sessions per subject) were chosen from the Photoface database to build the 'Photoface-10' (10 for the size of the gallery) database. Each session in the Photoface-10 database was augmented using the method described, rotating from -20° to 20° with step length of 10° along both X- and Y-axis (generating 25 augmented 3D images, one of which is the original) to generate the augmented database, storing 2000 sessions (200 sessions per subject). The resulting sessions were similar to those shown in Fig. 2.

2.2 Network Architecture

In deep 2D face recognition, current research mainly employs very deep neural networks, such as ResNet-101 [10]. However, here a shallower convolutional neural network is employed as a proof of concept allowing many more training events to be performed. There is no reason why our augmentation approach could not be employed using a far deeper network and this will be the focus of future work. The network has eight layers, where six are stacked convolutional layers for feature extraction, and the final two layers are fully connected, for classification. As in AlexNet, the training data of our network were resized to a fixed size of 224 X 224 and then fed into the first convolutional layer. The main data pre-processing performed was data augmentation as described above. Filters with a receptive field of 3 X 3 were used to convolute incoming data, while 'same' padding was carried out to maintain the layer size. All convolutional layers were attached with a non-linear 'Relu' activation function, eliminating all negative outputs. Spatial pooling was processed by three max-pooling layers with a size of (2 X 2) and stride of 2 pixels, shrinking the input image from 224 X 224 to 28 X 28. The last convolutional layer was followed by two fully connected layers having 1024 and 512 channels respectively, which were followed by the last soft-max layer. Regularization, such as drop out, was employed to avoid over-fitting.

3 Experiments and Results

3.1 Training a DCNN using the raw Photoface-10 database

To investigate if deep learning can effectively work in 3D face recognition, the network architecture described above was first trained using the raw Photoface-10 database on a NVIDIA 1080TI GPU, using Keras (version 2.1.3) and TensorFlow (version 1.5.0) backend. 48 sessions were randomly selected to be the training set, while the remaining 32 sessions formed the validation set. Such a distribution ensured every subject at least had 1 session in the validation set. Due to the relatively small data size, stochastic gradient descent with a batch size of 1 was used. Both training and validation accuracy was around 0.1 at the beginning of training. This is because the size of the gallery set was 10 and a random prediction gives an accuracy of 10%. At the end of training, while the training accuracy approached 100%, the validation accuracy still remained at around 70%. The best-trained model only achieved a validation accuracy of 75%. Both

training and validation accuracy drastically fluctuated during the training process, indicating that the network was not properly regularized and did not converge. The poor performance obtained in this experiment was not unexpected. Without sufficient training data, gradient descent was unable to converge the network. A shallow face recognition algorithm, Eigenface, was also experimented with on the raw Photoface-10 database. Five sessions for each subject in the raw database were used for training, while the remaining three sessions were used as the validation set. The validation accuracy of the Eigenface algorithm was only 70% on the raw Photoface-10 database. Therefore, the DCNN did not show a large advantage compared with the shallow Eigenface algorithm on the raw Photoface-10 database (only 5% improvement in accuracy). This result illustrates one of the major weaknesses of deep learning: such a data-hungry method does not apparently outperform a shallow method when large-scale training data is not available.

3.2 Training a DCNN with an augmented database

The augmented Photoface dataset (2000 sessions, 10 subjects), as described previously, was next used to train the same DCNN as before, with the same split of training and validation sets. Since here the network converged quickly, it did not need to be run for so many epochs. The entire training process only lasted for 60 epochs and the model with lowest validation loss was saved in case of over-fitting. The variation of training and validation accuracy during the training progress is shown in Fig. 3. As the training progressed, both training and validation accuracy increased rapidly and exceeded 90% after nearly 20 epochs of training. The first 100% validation accuracy occurred at epoch 27.

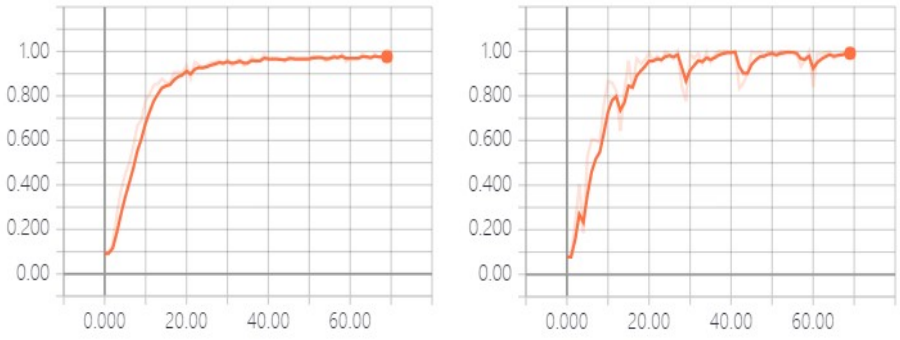


Fig. 3. The variation of training (Left) and validation (Right) accuracy during the training process; the horizontal axis refers to training epochs.

On the right in Fig. 3 it can be observed that once the validation accuracy reached 100%, it drops to nearly 90% three times in subsequent epochs. This could be explained if a minima meant the training accuracy did not achieve 100% at the same time as validation, so that back propagation still modified the model parameters due to training error,

causing degradation. However, such degradation can be self-corrected, as also shown in Fig. 3 – the degradations become increasingly small in size as the epochs increase. As training progresses, the decaying learning rate takes the model less far away from the minimum and thus it requires fewer epochs for backpropagation to bring it back. At the end of training, the validation accuracy of the best model saved in the training process achieved 100%.

The deep convolutional neural network (DCNN) trained on the augmented Photoface10 database performed best, with a 33.3% improvement over the same DCNN trained on the raw Photoface-10 database. Given this result, the effectiveness of the proposed data augmentation method was proved. By augmenting the database by rotating sample data to mimic pose variation, a deep neural network can effectively learn the non-linear mapping with good generalization and extract distinct features from training and validation data. In order to investigate if DCNNs were able to recognize grayscale facial data, grayscale facial images for NX, facial albedo and NY data, were extracted from the augmented Photoface-10 database. These experimental results were also excellent, with validation accuracy for the three types of facial grayscale data ('Grey' NX, 'Grey' albedo, 'Grey' NY) of 100%, 100% and 99.88% respectively. This shows that DCNNs can extract distinct features from 3D grayscale facial data as long as sufficient training data is provided. The effect of the number of augmentation samples was also investigated; and it was found that 100% validation accuracies were obtained when 4 or more augmented samples (per raw session) were employed. Interestingly, it was found that generating more artificial samples caused the network to converge earlier. It is thought that the smaller intra-variance of the additional samples may enable the neural network to learn the similar non-linear mapping within one epoch; however, more work is needed to be sure of the cause. Investigations were also undertaken to determine whether channel shifting is a valid data augmentation method. Although channel-shifting 2.5D images produces different image representations, if they were discriminable by a DCNN, the method could be valid for data augmentation. To verify this, data from both the raw Photoface-10 and the raw FRGC-10 database were augmented by shifting three channels of original 2.5D images (each original 2.5D image could generate 5 more artificial sessions), forming channel-shifting databases which were used to train the DCNN described above. The channel-shifting augmentation method achieved 98.96% and 99.48% validation accuracy respectively, showing that channel shifting is a valid data augmentation method for deep 3D face recognition.

3.3 Training a DCNN using the FRGC v2.0 and Photoface-50 databases

To check the ability of the data augmentation method to generalize, it was employed with two other databases. A new database, FRGC-10, was generated from the FRGC v2.0 database [8] using a similar method as with the Photoface-10 database. Each session in FRGC-10 generated 24 more sessions (in total, 2000 sessions for 10 subjects) with the same rotating parameters as in Photoface-10. Both raw and augmented

FRGC10 databases were used to train two DCNNs and the results repeatedly demonstrated the effectiveness of the proposed data augmentation method. The network trained by the augmented FRGC-10 database achieved 99% accuracy on the validation set, while the network trained by raw FRGC-10 database only obtained a validation accuracy of 56.25%. A new database, Photoface-50, was also built, by randomly selecting 50 subjects from the Photoface database. Due to time constraints, each subject in the Photoface-50 database only had 100 sessions that were augmented from 4 raw sessions for each subject, with similarly augmented hyperparameters as before. A DCNN was trained using the Photoface-50 database and the best-trained model obtained a validation accuracy of 99% at epoch 102. The conclusion from these tests was that the data augmentation method works well for a range of databases, including more extensive ones with larger numbers of subjects.

3.4 Transfer Learning

In addition to training a bespoke neural network using augmented 3D facial data, transfer learning is another method of coping with insufficient training data. Transfer learning is a powerful tool that transfers previously acquired knowledge from a solved problem to an unsolved but related one. Transfer learning removes the classification layers (usually fully connected layers) of a pre-trained DCNN and keeps the hidden layers as a fixed feature extractor for the new data [11]. Previous research has shown that the 2D feature extractor (hidden layers) of a pre-trained DCNN can be effectively used in 3D feature extraction for range images [12]. Here experiments were carried out to investigate if the pre-trained feature extractor of a DCNN can extract valid features from 2.5D images. A VGG-19 network, pre-trained on the ImageNet database, was used. All weights belonging to the hidden (convolutional) layers were transferred and frozen. To be comparable with the bespoke network trained in previous experiments, the last three fully connected layers, which were fine-tuned in this experiment, were in exactly the same configuration as before. A VGG19 network that was pre-trained on photographs performed excellently on three augmented databases, achieving 100%, 99.75% and 100% on Photoface-10, FRGC-10 and Photoface-50 databases respectively, which shows that the facial representation of the 2.5D surface normal data is very similar to that of the 2D RGB image. Therefore a 2D pre-trained VGG19 network can extract features from 2.5D facial surface normal images in a similar way to RGB images. This experiment again justified the effectiveness of the proposed data augmentation method, since a fine-tuned network trained on augmented databases achieved a significant increase in validation accuracy, achieving 7%, 31% and 47.5% validation improvements for Photoface-10, FRGC-10 and Photoface-50 databases respectively. Additionally, transfer learning exhibits an advantage in terms of convergence speed, where the use of transfer learning scaled the training process down by a factor of at least 3. Such a phenomenon makes sense because the pre-trained feature-extraction layers in the VGG19 network enable the network to extract useful information for classification from epoch 1, while at this stage the hidden layer in the bespoke network only outputted chaos and meaningless features.

4 Conclusion

Due to the lack of large-scale 3D face databases, a pose-based data augmentation approach was proposed to generate artificial 3D facial sessions. Facial point cloud models were created and then rotated in 3D space to mimic the challenge of insufficient pose variation in facial scans. Using the point cloud model, 2D images, such as facial grayscale (or even color) and albedo images, can be rotated as well. This method can generate hundreds of augmented pose-variated facial images for one raw session, which can enable deep convolutional neural networks (DCNNs) to address the pose variation challenge in face recognition. This augmentation method was explored using two public 3D face databases, Photoface and FRGC v2.0, by training a bespoke convolutional neural network and fine-tuning a pre-trained VGG19 network. Channel-based augmentation was also shown to be effective, indicating that feature extraction in DCNNs is channel-invariant. Transfer learning was found to provide competitive performance and advantages in training times. To summarise, the experimental results obtained are excellent, showing that the DCNN achieves an improvement in validation accuracy after training with the augmented training sets, from less than 80% to nearly 100%; which is comparable to the state-of-the-art and proves the effectiveness of the proposed data augmentation methods.

References

1. Ben Soltana, W., Huang, D., Ardabilian, M., Chen, L., and Ben Amar, C. Comparison of 2D/3D Features and Their Adaptive Score Level Fusion for 3D Face Recognition. 3D Data Processing, Visualization and Transmission (2010).
2. Soltanpour, S., Boufama, B. and Jonathan Wu, Q. A survey of local feature methods for 3D face recognition. Pattern Recognition, 72, pp.391-406 (2017).
3. Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. Proceedings of the British Machine Vision, 1(3):6 (2015).
4. Schroff, F., Kalenichenko, D. and Philbin, J. FaceNet: A unified embedding for face recognition and clustering, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), DOI: 10.1109/CVPR.2015.7298682 (2015).
5. Zafeiriou, S., Hansen, M., Atkinson, G., Argyriou, V., Petrou, M., Smith, M. and Smith, L., The Photoface database. CVPR 2011 WORKSHOPS (2011).
6. Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet classification with deep convolutional neural networks. Communications of the ACM, NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, Vol. 1, pp.1097-1105 (2012).
7. Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet classification with deep convolutional neural networks. Communications of the ACM, Volume 60 Issue 6, pp.84-90 (2017).
8. P. Jonathon Phillips, Patrick J. Flynn, Todd Scruggs, Kevin W. Bowyer, William Worek, Preliminary Face Recognition Grand Challenge Results', Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FGR'06) (2006).
9. Hansen, M., 3D Face Recognition Using Photometric Stereo. PhD. The University of the West of England (2012).

10. He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).
11. Shin Hoo-Chang, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers., Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning, IEEE Trans Med Imaging. May 2016; 35(5): 1285–1298 (2016).
12. Kim, D., Hernandez, M., Choi, J., and Medioni G. Deep 3D face identification. arXiv preprint arXiv:1703.10714 (2017).

Hybrid Gravitational Clustering with Centroids

Nathaniel Zeiger[1] and Raed Seetan[1]

[1] Slippery Rock University, Slippery Rock, PA 16057, USA

Abstract. Gravitational clustering is a category of methods for grouping unlabeled data points by modeling data as objects and simulating their movement under the force of gravity. While others have established the validity of this approach, this paper explores and evaluates a novel implementation by which clusters are formed. In this paper, we present a hybrid gravitational clustering algorithm that combines two different clustering techniques based on gravitational law and Newton's second law of motion, where points in a dataset are considered as physical objects in space. Our approach is divided into two parts: first, centroids treated as objects in feature space are distributed according to the gravitational force exerted on them by other points in the data set. Second, points in the dataset move according to the gravitational force exerted on them by the centroids. Clusters are formed by points merging with one another based on their proximity. The performance is measured on synthetic datasets against k-means, Birch, and OPTICS. This paper yields a reliable approach to finding centroids in a data set that can also provide an alternate interpretation of ambiguously grouped data. Our results demonstrate that our proposed approach can consistently recover clusters across various kinds of data sets, performs comparably well or better than other tested clustering algorithms, and is resistant to noise and outliers.

Keywords: Clustering, Unsupervised, Gravitational

1 Introduction

The goal of clustering analysis is to identify similarities among a collection of unlabeled data points and group them such that data points assigned to the same cluster have high similarity, while the similarity between data points assigned to different clusters is low [1]. What constitutes similarity depends largely on the problem domain and the nature of the dataset. Many clustering algorithms, such as *k-means* and *k-medoids*, measure similarity as the proximity of individual points among *k* points in feature space. This approach requires a value for *k* to be chosen beforehand. For datasets free of noise and with uniform, roughly circular groups, the resulting cluster allocations are likely to reflect this interpretation of similarity: that points within a cluster are similar because they each are closest to one of many representative points.

For some distributions of data points, measuring their distance among a few representative points does not yield an accurate interpretation of similarity. *K*-means is one technique which uses proximity to centroids to form clusters [2]. Fig. 1 shows the formed clusters of two datasets when using *k*-means algorithm. The second plot

illustrates the problem that can arise from using a few representative points to determine similarity. Algorithms which use this approach struggle to correctly allocate data points in the presence of noise or when points are grouped in asymmetrical patterns [3]. In these cases, similarity is better defined as a property which emerges from data points' relative proximity to one another. For the data set shown in the top-most image of Fig. 1, it can be seen that there exists at least two points such that all samples belonging to a group are closest to the same of the two points. In data set shown in the bottom image of Fig. 1, this is not the case; there are two discernable groups, but there does not exist two points such that all samples of one group are closest to one of the respective points.

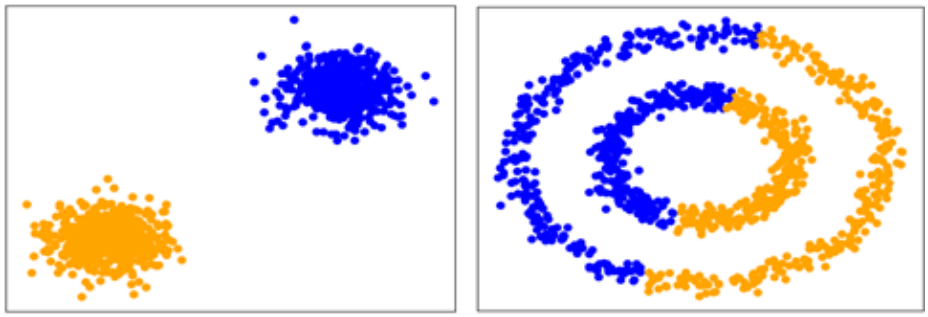


Fig. 1. Clusters formed by *k-means* on two data sets

Gravitational clustering is one such algorithm that uses this approach. Gravitational clustering algorithms consider data points or centroids as particles and use Newton's law of gravitation and second law of motion to move them in feature space [4]. Wright's proposed gravitational clustering algorithm [5] is a form of agglomerative hierarchical clustering, which finds clusters by iteratively improving partitions through a sequence of merges. Data points are treated as particles whose positions are updated with respect to the gravitational force they exert on each other. When two particles become close enough to one another, they merge to form a single cluster with combined mass. This process is repeated until only a single cluster remains. Wright's algorithm has a notably high time complexity, $O(n^3)$. It also inevitably merges all points to a single cluster, which after the fact a cut must be made at some point to yield a chosen number of clusters.

Several variations of Wright's original approach have been proposed in [6], [7], [8], [9], [10] the main goal of these approaches is to construct reliable clusters and reduce the computational complexity of the clustering process. In this paper, we present a hybrid gravitational clustering algorithm that combines two different clustering techniques based on gravitational law and Newton's second law of motion. Our proposed approach determines and uses only a few representative points which are used to calculate the force applied to objects in feature space. This study establishes that gravitational clustering remains effective on complex cluster shapes and in the presence of noise even when moving particles according to an approximation of the data set.

The remainder of this paper is structured as follows: Section 2 discusses the literature review works. Section 3 presents our proposed approach. Section 4 discusses the results

of our paper. Section 5 concludes the study.

2 Literature review

Since Wright, others have contributed several variations on his original proposal. Gomez et al. [11] devised an approach based on Wright's with significant improvements to its speed and computational complexity by updating particle positions with respect to another single random particle at each iteration instead of every other particle.

Gomez and Leon [6] have proposed a variation to handle circular data called "spherical randomized gravitational clustering" which modifies gravitational law to use cosine distance and uses geodesics to move points according to gravity. This algorithm was tested on two synthetic datasets, one with noise and one without. They demonstrated that the Newton gravitational model could be generalized to curved space and that this algorithm can find clusters in circular data in the presence of noise.

Bahrololoum et al. [7] use centroids distributed by gravitational force with respect to all points in a data set to determine clusterings. In this method, data points and the cluster centroids are considered fixed objects and movable objects which change their position in feature space. The performance of this algorithm was measured in a comparative experimental study with some well-known clustering algorithms on three datasets and several benchmark datasets from UCI, the results of which indicate its effectiveness.

N. Ilc and A. Dobnikar [8] uses an algorithm employing principles of gravitational law to retrieve information about connectivity gained in prototypes extracted from Kohonen's self-organizing map [12]. Each prototype extracted from the self-organizing map acts as a point in feature space from which information about connectivity can be derived from simulating gravitational force. These results of experiments performed on synthetic and real data sets show that this approach is able to discover complex cluster shapes.

Z. Wang et al. [9] have proposed a model in which each data point is considered an object with an associated local resultant force generated by its neighboring points, on the assumption that data points closer to cluster centers have a distinct difference in their local resultant fields compared to those at the boundaries of clusters. This approach was motivated by the observation that there are distinct differences between the local resultant forces of the data points close to the centers and boundaries of the clusters. Using this model, Wang et al designed a local gravitation clustering method and conducted several test cases on synthetic and real data to verify its applicability. Their results show that this technique achieves good performance on most of the data sets tested.

M. Sanchez et al. [10] developed an approach which groups data points using gravitational clustering and then "fuzzifies" the output clusters. This performance of this method on two data sets, one synthetic and one the Iris data set, are evaluated and compared against the fuzzy subtractive algorithm.

Our proposed algorithm differs from those developed in the aforementioned studies by approximating the data set with a number of centroids and moving data points

through feature space according to the gravitational force exerted by these centroids, rather than iterating over the entirety of the data set. The motivation for this approach is the hypothesis that gravitational clustering can perform effectively even when gravitational force is simulated with a small number of representative points, and when these points are not necessarily real data points.

3 Proposed approach

In this paper, we propose a two-phase hybrid approach of gravitational clustering using centroids. In the first phase, centroids are distributed across the feature space with a technique inspired by that proposed by Bahrololoum et al. In the second phase, data points are treated as particles which move according to gravitational force with respect to these centroids, set in the first phase, and are merged according to their proximity with other data points. This approach yields a capacity for complementary, alternative interpretive insights in how centroids are distributed, and also has an advantage of updating particle positions with locations informed by the entire dataset. In this section, we will introduce a related background that is used in our proposed approach, then we discuss the two phases of our proposed approach in details.

3.1 Background to illustrate the proposed approach

Our proposed approach uses a Newtonian model for motion and gravitation to form clusters, based on the technique developed by Gomez et al [11]. Each point in a data set is treated as a particle in space with a mass of 1. To conserve memory and improve performance, velocity is always considered to be the zero vector. Newton's motion laws and his law of gravitation are used to model how particles move through feature space. The force of gravity exerted from body x on another body y is modeled as the following equation:

$$F(t) = \frac{Gm_x m_y}{d(x(t),y(t))^2} \tag{1}$$

Where m_x and m_y are the masses of bodies x and y, $d(x(t), y(t))$ the Euclidean distance between them at time t, and G the universal gravitational constant (6.67×10^{-11}). Body y moves in towards x in the direction given by the vector:

$$d(t) = x(t) - y(t) \tag{2}$$

Using the directional vector, Equation 1 can be written as:

$$F(t) = \frac{Gm_x m_y}{||d(t)||^2} \tag{3}$$

Newton's second law of motion is given as:

$$F(t) = m_x a(t) \tag{4}$$

Where m_x is the mass of object x. Thus, the position of a body moving under the force of gravity exerted by another with mass m_x at time t can be derived by solving for acceleration a. In one-dimensional space, the speed and the acceleration of an object is defined by the following:

$$s(t) = \frac{dx(t)}{dt} \tag{5}$$

$$a(t) = \frac{ds(t)}{dt} = \frac{d^2x(t)}{dt^2} \tag{6}$$

Motion laws for n dimensions are vectorial extensions of the motion laws for one dimension. An object's velocity and acceleration are given by the following extensions of Equation 5 and 6 respectively:

$$v(t) = \frac{s(t)}{||\vec{d}||}\vec{d} \tag{7}$$

$$\vec{a}(t) = \frac{a(t)}{||\vec{d}||}\vec{d} \tag{8}$$

Where $\vec{d}$ is the directional vector, $\|\vec{d}\|$ is the magnitude of the directional vector, $a(t)$ the acceleration of the object, and $s(t)$ the speed of the object. When the acceleration is not constant, the velocity and position of the object can be approximated with the following:

$$v(t + \Delta t) = v(t) + \vec{a}(t)\Delta t \tag{9}$$

$$x(t + \Delta t) = x(t) + v(t)\Delta t + \frac{\vec{a}(t)\Delta t^2}{2} \tag{10}$$

The acceleration and acceleration vector caused by gravity are found with:

$$a(t) = \frac{Gm_y}{||\vec{d}(t)||^2} \tag{11}$$

$$\vec{a}(t) = \vec{d}\frac{Gm_y}{||\vec{d}(t)||^3} \tag{12}$$

The position equations for the movement of an object x due to the force of gravity exerted by object y are found using the estimated velocity and position formulas in Equation 13 and 14:

$$v(t+\Delta t) = v(t) + \vec{d}\frac{Gm_y}{||\vec{d}||^3}\Delta t \quad (13)$$

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \vec{d}(t)\frac{Gm_y\Delta t^2}{2||\vec{d}(t)||^3} \quad (14)$$

3.2 Phase one: centroid initialization

In this phase, *k* centroids are distributed across the feature space. The goal is to provide a centroid population that can approximate all of the data by iteratively moving the centroids by their gravitational attraction across all of the data points. However, this process causes all centroids to be pulled toward the same location. To prevent all of the centroids from converging to the same position, centroids repel themselves from one another. Their position is calculated using the same equation for updating position due to gravity, with a few differences:

$$x(t+\Delta t) = x(t) + \vec{d}(t)\frac{Gmy\Delta t^2}{||\vec{d}(t)||^2} \quad (15)$$

$$x(t+\Delta t) = x(t) - \vec{d}(t)\frac{Gmy\Delta t^2}{||\vec{d}(t)||^2} \quad (16)$$

Equation 15 is used to attract centroids to fixed data points; Equation 16 is used to repel centroids from one another. Reducing the order of distance prevents centroids from "rubber-banding", repeatedly attempting to converge to the same position when both are within a space with a high density of particles. The number of iterations required to obtain a sufficient approximation of the data increases with the number of samples in the dataset. The value chosen for k does not need to exactly equal the number of "true clusters" in the dataset, if it is known. Clusters are not formed by their proximity to the centroids; they are used as a means of approximating the data and updating particle positions accordingly in the subsequent phase of the algorithm. Figure 2. illustrates the paths that centroids travel from their initial positions to their final positions in this phase.

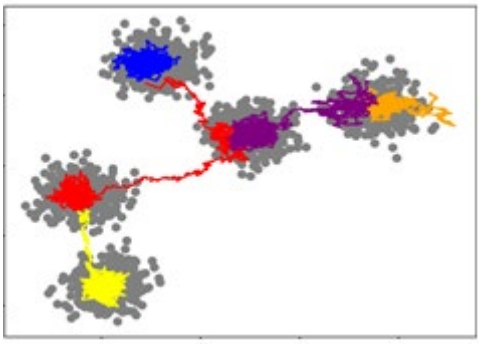


Fig. 2. Visualization of centroid initialization in phase one

Algorithm I illustrates the steps for this phase. Create k centroids in random positions (lines 1-2). Each particle representing a point in the data set remains fixed in place. At each of *M* iterations, where *M* is the number of points in the data set, move each centroid by the gravitational force of a random point in the data set (line 5) according to Eq. 15, then choose another random centroid and move the first by its gravitational force in the opposite direction of the other (line 6) according to Eq. 16.

Algorithm I for phase one:

```
1 for i = 1 to k:
2     c_i = random point ∈ x
3 for i = 1 to M:
4     for j = 1 to k:
5         ATTRACT (c_j, random point
6         REPEL (c_j, other random centroid)
```

3.3 Phase two: data points clustering

In the second phase, clusters are formed by points' proximities to one another using a disjoint set union-find data structure. When two points become close enough such that their distance is less than or equal to ε, they both remain individual points in the simulation and the union-find data structure is updated to indicate that these points belong to the same set. The term ε is calculated as the maximum inter-point distance in the data set multiplied by a chosen parameter γ.

Algorithm-II illustrates the steps for this phase. Each centroid remains fixed in place. At each iteration, select a random point and a random centroid (lines 1-3). Move the point toward the centroid by its gravitational force (line 4). Then, select another random point. If their distance is less than or equal to ε, merge the points (line 6) according to Eq. 14. If G remains constant, all points will eventually merge to a single position. To counteract this, a term Δ is introduced to slightly reduce G after each iteration (line 7).

Algorithm II for phase two:

```
for i = 1 to M:
    c = random centroid
    j = random data point
    ATTRACT (j, c)
    k = another random data point
    if DIST (j, k) ≤ ε then UNION (j, k)
    G = G * (1-Δ(G))
```

Data points which are isolated and relatively distant from most other data points, Outliers, are not likely to be merged with others or are more likely to be merged with other outliers. To filter out noisy data points, a parameter α is used to determine the minimum required size of a cluster. When clusters are recovered from the union-find structure, sets with less than the minimum required size are not considered as a valid cluster and its members are labeled as outliers.

4 Results and discussion

To test our proposed approach, we evaluated its performance on three labeled synthetic datasets, Figure 3, against k-means, Birch, and OPTICS clustering algorithms. In our experiment we ran each algorithm on each of the datasets twenty times, measured the adjusted rand index, homogeneity, completeness, and Fowlkes-Mallows score of each of the resulted formed clusters and averaged the results. The adjusted rand index measures similarity between ground truth class assignments and the predicted assignments. Homogeneity is a measure of whether each cluster contains only members of one class. Completeness measured whether all members of one class are assigned to the same cluster. The Fowlkes-Mallows score is the geometric mean of the precision and recall and is used as a metric which quantifies the similarity of two clusterings.

Fig. 3. Synthetic data sets used for testing

Each data set contained 1000 samples. The parameter values we used for gravitational clustering were $G_1 = 0.01$, $G_2 = 0.0001$, $\Delta = 0.0001$, $M_1 = 400$, $M_2 = 150{,}000$, $\gamma = 0.03$, and $\alpha = 0.02$, where G_1, G_2, M_1, and M_2 are the gravitational constants and number of iterations for phase one and phase two respectively. The parameters used for OPTICS for Set 1 and 2 were: minimum samples = 20, $\Xi = 0.05$, minimum cluster size = 0.1.

For Set 3, these parameters were: minimum samples = 5, Ξ = 0.035, minimum cluster size = 0.2.

The experiments results are shown in Table 1, the results demonstrate that our technique is able to perform well compared with the other clustering algorithms on the synthetic datasets. Our proposed approach showed the least variation in the metrics measured across all three datasets, indicating the robustness in forming clusters in various kinds of datasets. In Set 2, our algorithm had the highest rand index and f-score out of the four. This suggests that our algorithm is well-suited to finding clusters that are not uniform in shape. Visual analysis of our results corroborate that our approach is able to find clusters with irregular shapes and is resistant to noise and outliers, Figure 4.

Table 1. Performance comparison on synthetic data sets

Set 1	Rand index	F-score	Homogeneity	Completeness
Our proposed approach	0.9467	0.9644	0.8757	0.9660
K-means	1.0000	1.0000	1.0000	1.0000
Birch	1.0000	1.0000	1.0000	1.0000
OPTICS	1.0000	1.0000	1.0000	1.0000
Set 2				
Our proposed approach	0.9624	0.9809	0.8841	0.9841
K-means	0.2433	0.6213	0.1840	0.1840
Birch	0.3475	0.6899	0.3437	0.3182
OPTICS	0.8933	0.8532	0.9612	0.9428
Set 3				
Our proposed approach	0.9624	0.9809	0.8841	0.9841
K-means	0.9645	0.9763	0.9498	0.9495
Birch	0.5603	0.7678	0.9598	0.5612
OPTICS	0.5878	0.7128	0.5909	0.7128

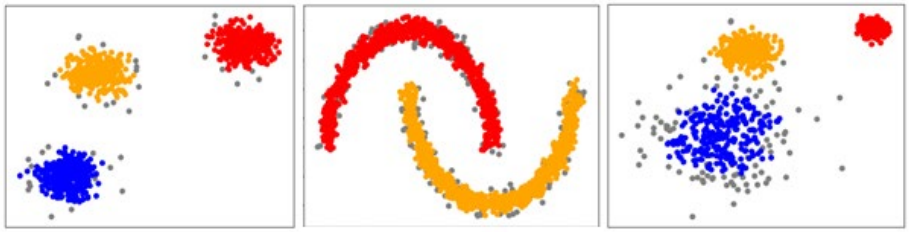


Fig. 4. Results of our proposed approach, data points in gray are considered outliers

Figure 5 shows an example of how centroids are used to approximate a synthetic data set. Figure 5(a) visualizes the movements of each centroid from its random starting position during each iteration of phase one. The ending locations of these centroids is where they remain fixed during phase two. Figure 5(b) shows the resulting cluster assignments from the second phase, with outliers marked in gray.

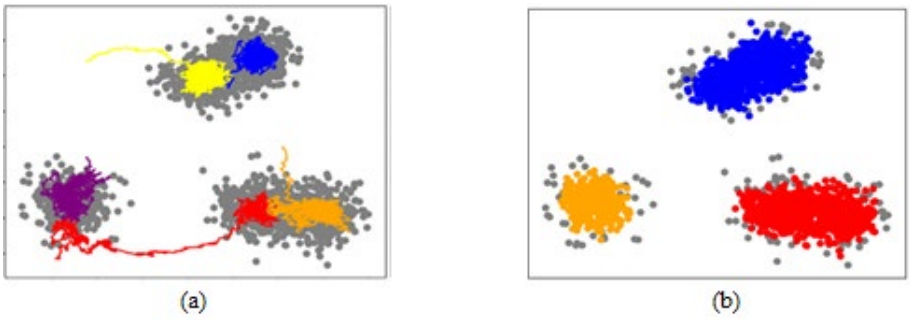


Fig. 5. : Visualization of our approach process, (a) shows the paths of centroid distribution in phase one, (b) shows the resulting clusterings from phase two.

Our algorithm is comprised of a series of operations which update random particle positions, of O (1), which are performed for each of M iterations. In the first phase, we found that a sufficient value for M tends to be less than or equal to N. In the second phase, in order to obtain an even distribution across all of the data, it is important that M be much larger than N. Thus, the time complexity of the second phase is O (M) where $M » N$. Recovering the clusters from the union-find structure is O (N); it scans an array containing an index of each point and separates them by their index.

5 Conclusion

We developed a gravitational clustering algorithm which combined two different variations of other clustering techniques based on gravitational law and Newton's laws of motion. We evaluated the performance of our algorithm against three other clustering algorithms on three synthetic data sets and showed how each phase of our approach can provide unique interpretations of the same dataset. Our algorithm can successfully find clusters in the presence of noise and outliers and works consistently well across various kinds of data sets. We believe that phase one of the technique we presented has utility in other applications. Future work will include exploring other potential applications of our method of centroid distribution and further testing our algorithm on real data sets.

References

1. M. H. Dunham, "Data Mining: Introductory and Advanced Topics, *Prentice Hall*, *Eaglewood Cliffs* 2003.
2. S. Khan, A. Ahmad, "Cluster center initialization algorithm for *K*-means clustering," *Pattern Recognition Letters*, vol. 25, Aug. 2004.
3. A. Kaur, P. Kumar, P.Kumar, "Effect of noise on the performance of clustering techniques," *IEEE Int. Conf. Networking and Info. Tech.*, Jun. 2010.
4. A. Aghajanyan, "Introduction to Gravitational Clustering," *Pattern Analysis and Machine Intelligence,* vol. 10, Feb. 2015.

5. W. E. Wright, “Gravitational clustering,” *Pattern Recognition,* vol. 9, Oct. 1977.
6. J. Gomez, E. Leon, “Spherical Randomized Gravitational Clustering,” *Proceedings of the 8th Alberto Mendelzon Workshop on Foundations of Data Management,* Jun. 2014.
7. A. Bahrololoum, H. Nezamabadi-pour, S. Saryazdi, “A data clustering approach based on universal gravity rule,” *Engineering Applications of Artificial Intelligence,* vol. 45, Oct. 2015
8. N. Ilc, A. Dobnikar, “Gravitational clustering of the Self-Organizing Map” *Adaptive and Natural Computing Algorithms*, 2011.
9. Z. Wang, Z. Yu, C. Chen, J. You, T. Gu, HS Wong, J. Zhang, “Clustering by Local Gravitation,” *IEEE Trans Cybern,* May 2018.
10. M. Sanchez, O. Castillo, J. Castro, A. Rodriguez-Diaz, “Fuzzy granular gravitational clustering algorithm,” *Annual Meeting of the NAFIPS*, Aug. 2012.
11. J. Gomez, D. Dasgupta, O. Nasraoui, “A New Gravitational Clustering Algorithm,” *SIAM Int. Conf. Data Mining,* 2003.
12. T. Kohonen, *Self-organizing maps*, Springer, Heidelberg, 2001.

Deep Learning Based Android Malware Detection Framework

Soumya Sourav[1], Devashish Khulbe[2] and Naman Kapoor[3]

[1] University of Texas at Dallas, Dallas, TX 75080, USA
[2] New York University, New York, NY 10003, USA
[3] Delhi Technological University, New Delhi, India 110042
sxs180011@utdallas.edu
dk3596@nyu.edu
naman.kapoor12@gmail.com

Abstract. With the development in the field of smartphones and ever growing base of Internet, various softwares are left prone to many malicious activities like pharming, phishing, ransomware, spam, spoofing, spyware, eavesdropping, etc. These threats have not spared the smartphones which are equally prone to them. In this work, we aim to detect these malwares with accuracy and efficiency. This being essentially a classification problem, we use various machine learning methods for this task. We observe that across models, Attention based Artificial Neural Networks (ANN), or broadly speaking, Deep Learning, are most suitable for this problem. Attention based ANNs are an amalgamation of accuracy and efficiency, the crux of our work. The accuracy achieved by our model is around 96.75%. Our model runs the test on Android Package Files (APKs) to determine whether a particular application is malicious or not by doing behavior analysis on android application under consideration.

Keywords: Malware detection · Deep Learning · Permissions, APK

1 Introduction

[1] The recent ubiquitousness of mobile phones for doing each and every task in day to day life because of there increased functionality has left them vulnerable to many new malware threats. Along with the nature of mobile devices, which are upgrading each day, the mobile threats are also upgrading with each passing day. The term mobile now includes all kinds of devices of IoT. Threats can be in the form of banking trojans, ransomwares, spywares, etc. which often leads to stealing private data and money from the user. Malware detection has continued to be one of the biggest challenges in today's technological world considering the fact that the resources available for doing the same are very limited. [17] According to McAfee malware analysis report 2017 the unprecedented rise of ransomwares is expected to continue in 2017. Google has removed more than 4000 apps from playstore in the past year without any information to the users. According to the

Telemetry data obtained by McAfee Mobile Threat Research, more than 500,000 mobile users still have these apps installed in their devices thus making them vulnerable to the threats these apps pose . [15] Kaspersky Mobile security products detected 1,319,418 malicious installation packages, 28,796 mobile banking trojans, 200,054 mobile ransomwares. TrojanBanker has become one of the most easiest and most used malware by fraudsters.AndroidOS.Asacub is one of the malwares most prominently used. The Q2 2017 results of the Kaspersky Lab were equally threatening with malware detection being at 1,319,148 which is twice the previous quarters data. In Q2 2017, Adware with a contribution of 13.31%, was the biggest source of malwares. The share increased by 5.99%. The majority of all discovered Installation packages are detected as AdWare.AndroidOS.Ewind.iz and AdWare. AndroidOS. Agent.n. Trojan-SMS malware (6.83%) ranked second in terms of the growth rate: its contribution increased by 2.15 percentage points. The problem of using machine learning based classifier to detect malwares using android permissions presents 3 main challenges: first, we need to extract an apps main features which will provide the base to classify them; second, important features should be identified and taken into account from the obtained dataset.; third, classification using the model. The first problem can be solved using various reverse engineering tools like androguard, apk tool, etc. which extracts an APK file to give its permissions, API calls and other related information about the file. We have used androguard in this project. The second problem is solved using Principal Component Analyses for Logistic Regression, Gradient Boosting and Naive Bayes and [4] sklearn.ensemble.ExtraTreesClassifier (ETC) for Neural Network. ETC is a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The third problem is solved using separate models out of which the Artificial Neural Network model that consists of an eleven-layered structure with 105 number of nodes in each layer gives the highest accuracy. This paper is organization augments understanding at every level: in Section II, we have discussed the related works in this field; in Section III, we have explained the working of our model; in Section IV, we have presented the components of our methodology; in Section V, we have discussed our experimental results; & finally, in Section VI, we have provided the references of our project .

2 Related Work

The machine learning methods for classification [1,2,3,4,5,8] has been very prevalent among data scientists and researchers. With some changes in their framework some encouraging results have been obtained. Another work [2] used the ensemble features to categorize among the malware apps. They have utilized the androguard as well for breaking down the android permissions along with mining the API calls and software/hardware features. The project is implemented in two phases: first one includes using separate features and the other one using the combined or the ensemble features. The accuracy was around 93% for the

ensemble technique. One other work [4,9] has used android permissions for their projects as well. However, k-nearest classifier [4] is used in one and random forest tree classifier [9] is used in another. Both are very robust and powerful machine learning algorithms. By analyzing the number of times each system call has been issued during execution of certain task, Crowdroid [6], a machine learning-based framework, recognizes Trojan-like malware. The trojanized version of an application is very different from its genuine application, since every time it is used, it issues different types of and different number of API calls. The project [7] makes use of parallel functioning of machine algorithms for storing different features. The algorithms used are Rule Based Classifier, Function Based Classifier, Tree Based Classifier, Probabilistic Classifier. All the features from these algorithms, after they are trained, are combined and a net result is obtained whether an app is malicious or not. The proposed method generates some good results, however the inclusion of four different algorithms surely speeds it down.

3 Methodology

3.1 Part 1: Data & Features

First phase of the work involves reading dataset, which includes permissions and API calls of various apps which are classified whether they are malwares or benign applications. Relevant features are then selected using two separate techniques: Principal Component Analyses (PCA) and a wrapper classifier by segregating features w.r.t. their importance. Irrelevant features are then removed for better generalization and better accuracy as well as reduced time involved in classification.

Dataset The dataset used for analysis is provided by Kaggle, a competitive platform where data miners and scientists can obtain numerous datasets and can compete in various challenges hosted by this domain only. Generally companies and researchers post some data on it and data miners compete to provide the best predicting accuracy on that data. The dataset contains features of 338 applications which are labelled as 'benign' or 'malicious'. This label is used to do feature selection and classification analysis by supervised learning. The dataset comprises of a .csv file. The top row contains all the permissions. The presence of a specific permission in an application sample is indicated by 1 and 0 if it is not present.

Imbalance problem - the situation when the contents of each class are not proportionate- is a very important aspect when using binary classifiers for the detection of malicious code. In our case the absence of malwares and legitimate files in equal proportions causes the imbalance problem. It is need of hour to continuously update the training set with new malicious files for making the classifier better. This is really an important aspect in maintaining such a framework. The dataset is updated to accommodate for new samples using androguard, an opensource project to extract features from APKs. This solves for the imbalance

problem by maintaining the proportion of benign and malicious entries to be equal.

Features A feature is a significant estimable property or characteristic of an event under observation. Each attribute in the dataset set signifies a unique feature and each entity signifies a data sample (an android application in this case). A feature in this project corresponds to a permission specified by an application. The dataset contains 330 features. The feature vector contains feature data in binary form where '1' indicates the presence of a permission and '0' indicates absence. The malware analysis approach implemented in this paper uses static android features extracted from the android app which are used to determine if app contains any malicious permissions which depends on a trained classification model. Androguard does the job of extracting features to train the ANN model combination of malware and benign APKs. Three categories of features are used for learning phase: 1. App Permissions 2. API calls 3. Standard OS and framework commands. API calls are the calls which are made to the server in the name of an application using a SDK or an API. Android permissions are requests or permissions acquired by the apps in order to use certain system data and features to maintain security for the system and user.

Table 1. Overview of the features extracted from the apps.

Type	Features
API call related	abortBroadcast, getDeviceId, getSubscriberId, getCallState, getSimSerialNumber, android.provider.Contacts, android.provider.ContactsContract; HttpUriRequest, SMSReceiver, bindService, onActivityResult, LjavaxCryptoSpecSecret, DexClassLoader, getNetworkOperator, getSimOperator
Command related	.apk; pmsetInstallLocation; pminstall; GET-METADATA; GET-RECEIVERS; GET-SERVICES; GET-SIGNATURES; GET-PERMISSIONS
Permissions	android.permission.UPDATE-LOCK; android.permission.USER-ACTIVITY; android.permission.VIBRATE; android.permission.WAKE-LOCK; android.permission.WRITE-CALENDAR; android.permission.WRITE-CALL-LOG; android.permission.WRITE-CONTACTS; android.permission.ACCESS-FINE-LOCATION

Feature Extraction Overfitting is an issue where the model instead of describing the underlying relationship, describes noise and random error. An overfitted

model tends to perform very poorly on predictive tasks as it overreacts to minor fluctuations in training data. This is a very common problem with every machine learning model. Huge and complex datasets affect training models by reducing the efficiency of the classification problem by using unnecessary number of features resulting in increased computation time. Hence extraction of important features is a key part before modeling. We use two methods for selection of important features:

Principal Component Analysis Principal Component Analysis (PCA) [12] is an important technique to deal with multicollinearity in the data and eliminating redundant variables. It drops the "least important" variables while still retaining the most valuable parts of all of the variables. As an added benefit, each of the new variables after PCA are all independent of one another. In our data set, the original feature space of 330 variables is reduced to 23 variables after using PCA. This transformed feature space accounts for 90% of the variance in the original space.

ExtraTreeClassifier ExtraTreeClassifier is an amalgamation of a search technique, which selects the features, and an evaluation measure, used to score the feature subsets. The algorithm varies in complexity with some being as simple as testing all the possible subsets of selected features and then winnowing the one with minimal error rate. One of the most influential factor for the algorithm is the choice of evaluation metric and these evaluation metrics are the one's which create a distinction among three main feature selection algorithm: filters, embedded methods and wrappers. This technique makes use of wrapper method to figure pout the accuracy of the selected features' subsets. By counting the number of mistakes our model makes while predicting on the hold-out set, the accuracy can easily be determined. Wrapper methods tend to be very intensive when it comes to computations. However the best results on a model can be obtained by this method only.

3.2 Part 2: Modeling

Second part involves training the machine learning models using the processed dataset and saving weights and the trained model. We use Logistic Regression, Naive Bayes and Gradient Boosted Trees along with the artificial neural network and compare their performance on the test data. We use the truncated feature space after the PCA and ExtraTreeClassifier segregates the useful variables. The test dataset is dynamic and keeps on changing whenever the program is run. It consists of around 35-40 permissions belonging to the original dataset with almost 330 permissions and generated on 398 applications both benign and malicious. This models after being trained, saves the weights and the model. These weights and model are then called which breaks individual apps to be checked and then the neural network model decides whether the app is malicious or benign.
The crux of our work is developing the Artificial Neural Network for this problem

and compare it with other off the shelf techniques. Thus, we focus on the results of ANN for major parts of the modeling and results section.

Classification Methods We begin with off the shelf models for the classification task. We use Logistic Regression, Naive Bayes, Random Forests and Gradient Boosting models before moving onto deep neural networks. These methods are fairly well known and have proven to perform good for binary classification problems. The problem with these approaches, however, is that they might not be able to learn complex non-linearity which might be in the data. To tackle this, we delve into developing Artificial Neural Network for the task.

For the Artificial Neural Network we add an attention layer to enhance the prediction capability of the neural network. The neural architecture has 12 Dense layers. In between every third Dense layer a dropout layer was added to reduce the overfitting of the neural networks. The dropout layer drops a certain percentage of the neurons in random. These neurons don't receive any updates and don't contribute to the output at all. The attention network [13, 14] is used in order to make the neural network capable of attending to a subset of the inputs which increases the space of functions which can be approximated by the neural network and makes it possible to look into entirely new use-cases which enhances the neural net's capability to converge to a global minimum. The attention network generally assigns the scores to the input features using the softmax function.

$$f(z)_{\mathrm{j}} = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{1}$$

where $\mathrm{f_j}$ is the output of attention layer and $\mathrm{e_j}$ is the output previous Dense layer. This output is the probability distribution which is combined with the Dense layer output which gives the importance of each feature for the output prediction. Different methods for combining the output layer and probability distribution can be used like Dot product, Scalar Multiplication or Bilinear combination. For the final training procedure of the PKG and PCB model, a batch size of 32 was fixed after doing cross validation with the optimizer "Adam" was used.

4 Results

For analyzing the performance of our models, we decided to check their accuracies on the test data. Checking the accuracy makes sense as the distribution of positive and negative records in our data are equal in proportion. The training:test data proportion is set to be 75:25.

On comparing the accuracy across all models, ANN, due to its complex structure, is able to perform better than all of the other classification techniques. The validation accuracy is found to be 95%. The problem of classification could be countered with many machine learning algorithms like SVM or regression. However, the main advantage Neural Networks or the multilayer perceptrons hold

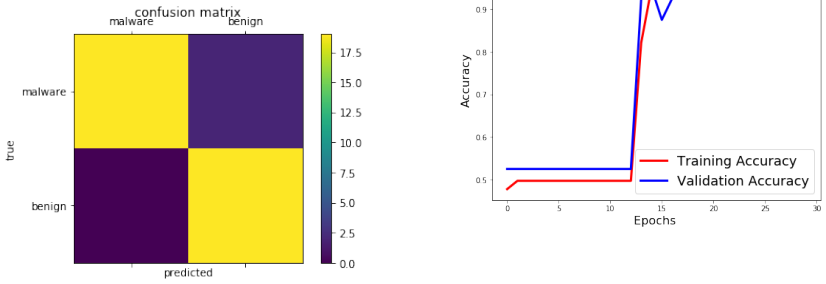
over these algorithms are the fixed number of layer size. The neural networks are parametric models while the machine learning algorithms are not i.e. in neural networks, there are a number of hidden layers depending upon the number of features. The outputs of neural networks can be multiple and they were tested to be faster than the machine learning algorithms.

Table 2. Classification accuracy across models

Name	Accuracy(%)
Naive Bayes	88.3
Random Forests	90.8
Gradient Boosting	91
Logistic Regression	93.6
Attention Based ANN	95.3

We also check the performance of our ANN model with respect to its predictions concerning the individual classes through a confusion matrix.

Fig. 1. Performance of ANN model: Confusion matrix (left) and accuracy curve (right)



Precision is a common metric to judge the performance of classification models. It is essentially the fraction of relevant examples classified correctly. With the ANN model, the precision score is found to be 0.90476.

$$ANNPrecision = \frac{TruePositive}{TruePositive + FalsePositive} = 0.90476 \tag{2}$$

5 Conclusion

We introduced a new Deep Learning based method to predict malwares in Android applications which performs significantly better than traditional off the shelf machine learning techniques. Due to the small size of the data we worked

on, this new method is yet to be tested on bigger and more complex data with more features and across various platforms. This can be a possible scope for future work. Also, we fed only the significant features from our original data into the models' inputs. There can be further scope in finding which features are important predictors for this task. Furthermore, we think malware detection is a fairly important area of study and different approaches and data sets still need to be explored in this domain.

References

1. Justin Sahs, Latifur Khan. "A Machine Learning Approach to Android Malware Detection", European Intelligence and Security Informatics Conference-2012
2. A.M. Ashwini, P. Vinod. "Android malware analysis using ensemble features", Springer international publishing Switzerland 2104
3. Nikola Milosevic, Ali Dehghantanha, Kim-Kwang Raymond Choo. "Machine learning aided Android malware classification", Elsevier ltd, 2017
4. Naser Peiravian, Xing Quan Zhu. "Machine learning for android malware detection using permission and api calls", IEEE 25th international conference on tools with artificial intelligence 2013
5. I. Burguera, U.Z., Nadijm-Tehrani. "Crowdroid: Behavior- Based Malware Detection System for Android", SPSM'11, ACM(October 2011)
6. Suleiman Y. Yerima, Sakir Sezer, Igor Muttik. "Android Malware Detection Using Parallel Machine Learning Classifiers",8th International Conference on Next Generation Mobile Applications, Services and Technologies, (NGMAST 2014), 10-14 Sept., 2014
7. Zami Aung, Win Zaw. "Permission-based android malware detection", international journal of scientific and technology research, vol. 2, issue 3 , 2013
8. Benjamin VA, Chen. "Machine learning for attack vector identification in malicious source code", 2013 IEEE international conference on intelligence and security informatics (ISI). IEEE; 2013. p. 21–3
9. Boxall A. The number of smartphone users in the world is expected to reach a giant 6.1 billion by 2020; 2015 http://www.digitaltrends.com/mobile/ smartphone-users- number- 6- 1- billion- by-2020/
10. Schmidt A-D, Clausen JH ,Camtepe A ,Albayrak S. "Detecting symbian os malware through static function call analysis", 4th international conference on malicious and unwanted software (MALWARE), 2009. IEEE; 2009. p. 15–22
11. Alam MS, Vuong ST. "Random forest classification for detecting android malware", 2013 IEEE and internet of things (iThings/CPSCom), IEEE international conference on and IEEE cyber, physical and social computing, green computing and communications (GreenCom). IEEE; 2013. p. 663–9
12. Wold, S., Esbensen, K. and Geladi, P., 1987. Principal component analysis. Chemometrics and intelligent laboratory systems, 2(1-3), pp.37-52.
13. https://skymind.ai/wiki/attention-mechanism-memory-network
14. http://akosiorek.github.io/ml/2017/10/14/visual-attention.html George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.
15. Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.

Predicting resource utilization in Cloud using Multivariate CNN LSTM model

Soukaina Ouhame[1] and Youssef Hadi [1]

1Department of computer science, Faculty of science, Ibn Tofail University
Soukaina.ouhame@uit.ac.ma, hadi@uit.ac.ma

Abstract. In cloud computing services, the Infrastructure (Iaas) is a service model that provides virtual computing resources such as : networking, hardware, and storage services as needed to users. However, cloud-hosting initialization takes several minutes delay in the hardware resource allocation process. To resolve this issue we need to predict the future amount of computing resources and allocate them before being requested. Cloud data centres can dynamically scale resources to decrease energy consumption while maintaining a high quality of service. In this paper, we propose a CNN-LSTM model for predicting multivariate workload, which are: CPU, Memory, Disk and Network usage. Firstly, the input data are analysed by the Vector Auto Regression (VAR) regression method, which filters the linear interdependencies among the multivariate data. Then the residual data are computed and entered to the Convolutional Neural Network (CNN) layer, which extract complex features of each of the Vm usage components, and after to the Long Short Term Memory (LSTM) neural network, which is suitable for modelling temporal information of irregular trends in time series components. The proposed model is compared with other predictive models. Results of experiments show superior efficacy of the proposed method over the other hybrid models.

Keywords: Multivariate prediction, CNN, LSTM, Cloud computing, Infrastructure, Vector Auto regressive, VAR

1 Introduction

Infrastructure as a service provides flexible and fast IT resources on demand. The majority of cloud providers offer scalable services that automatically provide computer resources (such as CPU, memory, and storage). However, the scaling time to initialize a new virtual machine mainly introduces a delay of several minutes. To reduce scaling time, it is important to fix the exact amount of resources in advance. Consequently, predicting Vm utilization is the key solution to solve this problem.

Predicting resource usage is an important tool for effective planning and to aid in counteracting future uncertainty. Many prediction methods have been used in the literature, Jiang et al [1] explains that the behavior of web and data center workloads can be modeled through time series models. Islam et al [2] uses Neural Network (NN) and

Linear Regression (LR) algorithms and the sliding window technique in order to develop a new workload prediction strategy.. Calheiros et al [3] uses the Auto Regressive Integrated Moving Average (ARIMA) model to predict cloud workload for Software as a Service (SaaS) providers. Zhang [4] proposes a hybrid approach to time series forecasting using both ARIMA and ANN models.

Recently, deep neural networks have been intensively used in workload prediction. In [5], the authors indicate that the LSTM model could solve the issues faced by cloud systems, as it is fragile and costly in the event of issues such as dynamic scaling of resources and energy consumption. The authors state that if it would be possible to determine the precise future workload of a server, resources can be adjusted according to demand and thus maintain both quality of service and reduce energy consumption.

We choose CNN-LSTM for resource utilization prediction in Cloud, CNN is used to remove noise and to take into account the correlation between multivariate variables, and LSTM models temporal information and maps time series into separable spaces to generate predictions. In this paper, we propose a CNN-LSTM neural network combining CNN and LSTM to predict Vm utilization. Vm utilization is multivariate time series that is recorded over time, including spatial information among variables and irregular patterns of temporal information [5]. We propose a CNN-LSTM model for predicting resource usage metrics, which are CPU, Memory, Disk and Network usage. Firstly, the input data are analysed by the VAR regression method to filter the linear interdependencies among the multivariate data. Then the residual data are computed and entered to the CNN layer, which extract complex features of each of the Vm usage components, and after to the LSTM, which models temporal information of irregular trends in time series components and generates the predictions.

The main contributions of this paper are as follows:

• We provide the fundamental definitions and necessary notions for building each of the CNN and LSTM models.

• We present the multivariate workload prediction algorithm.

• To conduct experiments and evaluate the proposed method, we use real world workload traces of GWAT-12 Bitbrains service provider. The paper is organized as follows. Section II presents the proposed method. The experimental results and analysis are shown in Section III. Finally, Section IV concludes the paper.

2 Proposed method

Our multivariate time series data is composed of two portions, the linear and the nonlinear portion. Thus, we can express as follows:

$$x_t = L_t + N_t + \mathcal{E} \qquad (1)$$

L_t Represents the linearity of data at time t, while N_t signifies nonlinearity. The $\mathcal{E}$ value is the error term.

Firstly, the multivariate time series x_t is analysed by the VAR model, which captures the linear trends. The residuals of the model (the nonlinear part N_t) contain spatial and temporal information:

$$N_t = S + T \qquad (2)$$

The spatial features are extracted by the CNN model then entered as inputs to the LSTM model, which is appropriate for modelling temporal information and generates the final predictions.

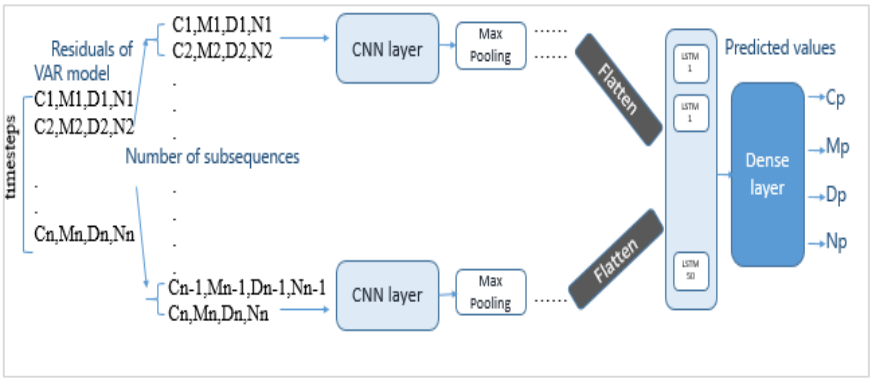


Fig. 1. The schema of the proposed model.

First, we introduce some background concepts of the two models. After that, we describe the proposed model for multivariate workload prediction in the cloud.

2.1 Vector Autoregressive model

VAR models introduced by Sims [6] is a univariate model extension for predicting multivariate time series. The structure is that each variable is a linear function of past lags of itself and past lags of the other variables.

$$y_1(t) = a_1 + w_{11}y_1(t-1) + w_{12}y_2(t-1) + w_{13}y_3(t-1) + w_{14}y_4(t-1) + e_1(t-1) \quad (3)$$

$$y_2(t) = a_2 + w_{21}y_1(t-1) + w_{22}y_2(t-1) + w_{23}y_3(t-1) + w_{24}y_4(t-1) + e_2(t-1) \quad (4)$$

$$y_3(t) = a_3 + w_{31}y_1(t-1) + w_{32}y_2(t-1) + w_{33}y_3(t-1) + w_{34}y_4(t-1) + e_3(t-1) \quad (5)$$

$$y_4(t) = a_4 + w_{41}y_1(t-1) + w_{42}y_2(t-1) + w_{43}y_3(t-1) + w_{44}y_4(t-1) + e_4(t-1) \quad (6)$$

Where $y_1(t)$,$y_2(t)$, $y_3(t)$ and $y_4(t)$ are the CPU, Memory, disk and network usage at moment t , $y_1(t-1)$, $y_2(t-1)$, $y_3(t-1)$ and $y_4(t-1)$ are the CPU, Memory, disk and network usage at moment t-1 (here the lag value is 1). a_1, a_2, a_3 and a_4 Are the constant terms,

, , ,..etc. are the coefficients, and e_1,e_2, e_3 and e_4 are the error terms. Before we can estimate a bivariate VAR model for the two series, we must specify the order p.

VAR order selection

The most common approach for model order selection involves selecting a model order that minimizes one or more information criteria evaluated over a range of model orders; Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) or Hannan-Quinn Criterion (HQC). In this paper, we resolve to use the AIC metric to estimate parameters

$$AIC = -2ln(\hat{L}) + 2k \qquad (7)$$

The $ln(\hat{L})$ notation is the value of the likelihood function, and k is the degree of freedom, that is, the number of parameters used. A model that has a small AIC value is generally considered a better model. The residual values are computed and entered to the subsequent CNN-LSTM model. As the VAR model has identified the linear trend, the residual is assumed to comprise the nonlinear features.

$$x_t - L_t = N_t \qquad (8)$$

2.2 CNN-LSTM model

Convolutional neural network

The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional or with three-dimensional data.

In the time series forecasting problem, A 1D CNN is capable of reading across sequence input and automatically learning the salient features.
A one-dimensional CNN is a CNN model having a convolutional hidden layer that operates over a 1D sequence. This is followed by a second convolutional layer in some cases, such as very long input sequences. Equation (9) is the result of the vector y_{ij}^1 output from the first convolutional layer.

$$y_{ij}^1 = \sigma(b_j^1 + \sum_{m=1}^{M} w_{m,j}^1 x_{i+m-1,\ j}^0) \qquad (9)$$

Where x_{ij}^1 is the input vector, b_j^1 represents the bias for the j[th] feature map, w is the weight of the kernel, m is the index value of the filter, and σ is the activation function like ReLU. Then the convolutional layer is followed by the pooling layer whose job it is to distill the output of the convolutional layer to the most salient elements.

The pooling layer reduces the space size of the representation to reduce the number of parameters and network computation costs. The max-pooling used for resource usage

prediction uses the maximum value from each neuron cluster in the previous layer. This also has the effect of adjusting overfitting. Equation (10) represents the operation of the max-pooling layer. T is the stride that decide how far to move the area of input data, and R is the pooling size of less than the size of the input y.

$$p_{ij}^{1} = maxy_{r\epsilon R}^{1} i \times T + r.j \qquad (10)$$

The convolutional and pooling layers are followed by the LSTM layer that interprets the features extracted by the convolutional part of the model. A flatten layer is used between the convolutional layers and the LSTM layer to reduce the feature maps to a single one-dimensional vector.

Long Short Term Memory neural network

LSTM, which is a lower layer of CNN-LSTM, stores time information about important characteristics of power demand extracted through CNN. LSTM provides a solution by preserving long-term memory by consolidating memory units that can update the previous hidden state. This function makes it easy to understand temporal relationships on a long-term sequence. The output values from the previous CNN layer are passed to the gate units. The LSTM network is well suited for predicting power demand by addressing explosive and vanishing gradient problems.

The LSTM cell comprises four interactive neural networks, each representing the forget gate, input gate, input candidate gate, and the output gate. The forget gate outputs a vector whose element values are between zero and one. It serves as a forgetter that is multiplied to the cell state from the former time step to drop values that are not needed and keep those that are necessary for the prediction [5].

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \qquad (11)$$

The σ function, also denoted with the same symbol in Fig.2, is the logistic function, often called the sigmoid. It's the activation function that enables nonlinear capabilities for the model.

$$\sigma(X) = \frac{1}{1-e^{-x}} \qquad (12)$$

In the next phase, the input gate and the input candidate gate operate together to render the new cell state C_t, which will be passed on to the next time step as the renewed cell state. The input gate uses the sigmoid as the activation function and the input candidate utilizes the hyperbolic tangent, each outputting i_t and C'_t. The i_t selects, which feature in C' should be reflected in to the new cell state C_t.

$$i_t = \sigma(W_i,[h_{t-1}, x_t] + b_i) \qquad (13)$$

$$C_t = \tanh(W_c, [h_{t-1}, x_t] + b_c) \quad (14)$$

The tanh function in Fig.2 is the hyperbolic tangent. Unlike the sigmoid, which renders value between zero and one, the hyperbolic tangent outputs value between -1 and 1[5].

$$\tanh(X) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (15)$$

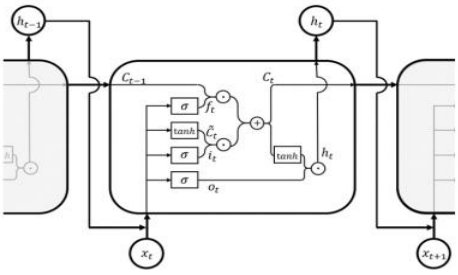


Fig. 2. Inner structure of a Long Short-Term Memory cell [5]

Finally, the output gate decides what values are to be selected, combining O_t with the tanh applied state C_t as output h_t. The new cell state is a combination of the forget-gate applied former cell state C_{t-1} and the new tanh applied state C_t .

$$o_t = \sigma(W_o, [h_{t-1}, x_t] + b_0) \quad (16)$$

$$C_t = f_t . C_{t-1} + i_t . C'_t \quad (17)$$

$$h_t = o_t . \tanh(C_t) \quad (18)$$

The cell state C_t and h_t output will be passed to the next time step, and will go through a same process.

The proposed CNN-LSTM model

Algorithm 1: VAR model fitting algorithm

Input: Y_1: CPU usage time series
Y_2: Memory usage time series
Y_3: Network usage time series
Y_4: Disk usage time series
P_{max}: the max lag order
Output: Residual: residual values of the two series
1. **If** Y_1, Y_2, Y_3 and Y_4 are not stationary **then**

2. Y_1=*Difference* (Y_1)
3. Y_2=*Difference* (Y_2)
4. Y_3=*Difference* (Y_3)
5. Y_4=*Difference* (Y_4)
6. **End If**

Grouping the two time series into an unique multivariate time series ***Y***

7. Y= *concatenate* (Y_1, Y_2, Y_3, Y_4)

Convert into input/output with the percentage of 80%

8. Train, Test = *divide* (Y, 0.7)
9. *Select_order* (maxlags = P_{max})
10. P_lag = order lag with least AIC value
11. Mfit = fit VAR (P_lag)
12. **for all** data **in** *Test* **do**
13. Residual = Test - *predict* (Train, Mfit)
14. **Return** Residual.

The input of the algorithm are the CPU, Memory, Network and Disk usage time series, which are formed using the historical data of the workload. The stationarity of each time series is checked by using the augmented Dickey–Fuller (ADF) test; if they are not stationary, we differentiate them.

In the following step, we select the lag order with the least value for model fitting, and then we compute the residual values.

Algorithm 2: CNN-LSTM model training algorithm

Input: Residual: Residual values of the VAR model
N_step: the lag step between each input and output

Output: trainPred**,** testPred: the predicted train and test data of the multivariate time series.

{*Phase1: Data preprocessing*}

1. Normalize Residual data
   Convert into input/output with the percentage of 80%
2. train_cl, test_cl = *divide* (Residual, 0.75)
3. X_train, y_train = *split* (train_cl, N_step)
4. X_test, y_test = *split* (test_cl, N_step)
5. Reshape X_train et X_test into (samples, subsequences, timesteps, features)

{*Phase2: Determine model parameters*}

6. **Define** *model*
7. **add** TimeDistributed**(**Conv1D (filters = 64, kernel_size=1, activation = 'relu', input_shape = (None, N_steps, n_features)))
8. **add** TimeDistributed**(**MaxPooling1D (pool_size=2))

```
        add  TimeDistributed (flatten())
        add  LSTM (units = 50, activation = 'relu')
        add  Dense (n_features = 4)

    {Phase3: Model fitting & estimation}
    Repeat
     Forward_propagate  model with X_train
     Backward propagate model with y_train
     Update model parameters
      MSE, MAE = evaluate_model (X_train, y_train)
        If MSE converged:
    End Repeat
    MSEt, MAEt =evaluate_model (X_test, y_test)

    {Phase4:  Model prediction}
      trainPred = predict (X_train)
      testPred   = predict (X_test)
  return trainPred, testPred
```

The CNN-LSTM prediction algorithm works in four main phases : Data preprocessing, fixing model parameters, Model fitting and estimation, and model prediction.

As cited before, the residual values calculated by the algorithm1 are entered to the CNN-LSTM model.

We use four time steps; every sample is split into a pair of subsequences. The CNN model can interpret every sub-sequence and therefore the LSTM can piece along the interpretations from the subsequences. As such, we will split every sample into two subsequences of two times per subsequence.
The CNN are defined to expect two time steps per subsequence with four options. the whole CNN model is then wrapped in TimeDistributed wrapper layers so it are often applied to every subsequence within the sample. The results are then interpreted by the LSTM layer, that uses fifty neurons or blocks, and eventually the dense layer outputs the prediction.

The ReLu (Rectified Linear unit) activation function is used for CNN layer and LSTM blocks.

$$f(x) = x^{+} = \max(0, x) \qquad (19)$$

Where x is the input to a neuron.

The problem of vanishing gradient can be greatly reduced using the ReLU family of activation functions.

ADAM optimization algorithm is used for stochastic gradient descent for model training. The network is trained for 100 epochs with batch size of 1.

3 Experimental evaluation

The current section presents the experimental evaluation of the proposed method, including, dataset collection, experimental results and the evaluation of the efficiency of the proposed method.

3.1 Experiment dataset

The dataset contains the performance metrics of 1,750 virtual machines in the Bitbrains distributed data center. We choose the first trace; fastStorage: the trace consists of 1250 virtual machines connected to storage area network (SAN) devices. Each file consists of a set of lines; each line represents an observation of the performance metrics of a virtual machine every 300 milliseconds since 1970-01-01.

We have selected 4000 observations for our workload prediction model.

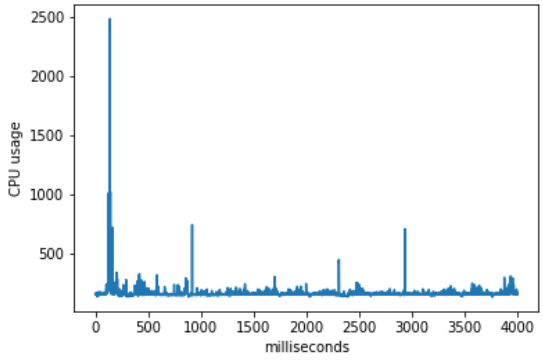


Fig. 3. CPU usage by milliseconds

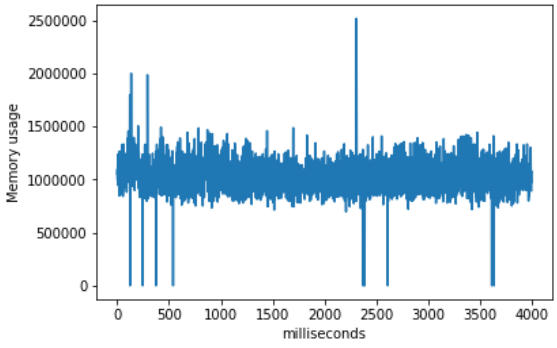


Fig. 4. Memory usage by milliseconds

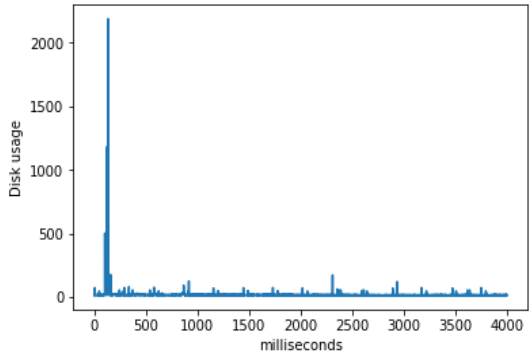


Fig. 5. Disk usage by milliseconds

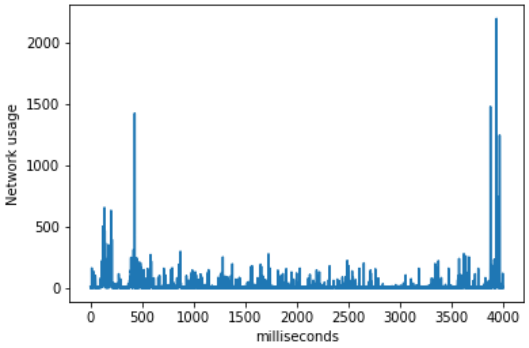


Fig. 6. Network usage by milliseconds

3.2 Analysis of the results

The resulting time series Y1 and Y2 are stationary, then, we do not need to perform a difference operation. In the first step, 70% of the multivariate time series Y is used for training and 30% for testing as indicated in the algorithm1. By choosing Pmax = 4 as the max lag order we have the following table (Table 1).

Table 1. VAR model order selection

	AIC	BIC
0	38.60	36.61
1	36.31	36.32
2	36.05	36.08
3	36.00*	36.04*
4	36.00	36.05

Where BIC stands for Bayesian Information Criterion, which also estimates the quality of a model. As the third lag order has the least value of AIC then it will be used for fitting the VAR model. In the second step, the residuals are calculated, 75% of data is used for training and 25% for testing.

To evaluate our model, the Mean Squared Error (MSE), Root Mean Squared Error (RMSE) values, and the Mean Absolute Error (MAE) values of the prediction were calculated.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 \quad (20)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y'_i| \quad (21)$$

Where y_i the output value and y'_i the predicted value

The MSE learning curve of both of the train and test data are close to each other, the same for the MAE learning curve(Figure 7,8).

The MSE values of train and test data have small variations, which means the model has been generalized adequately (Table 2).

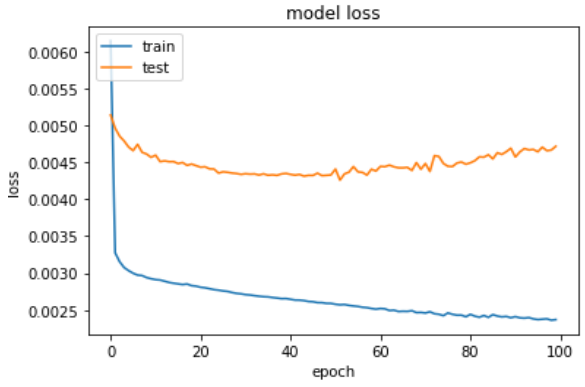


Fig. 7. MSE learning curve

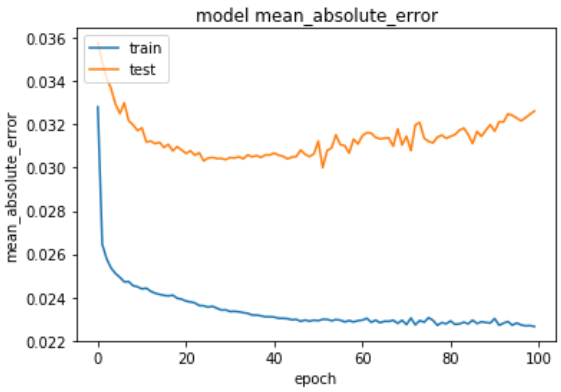


Fig. 8. MAE learning curve

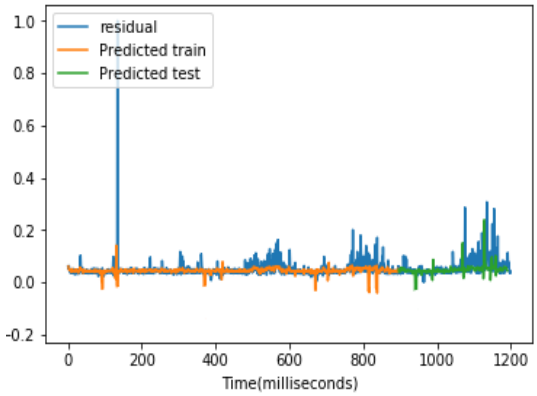


Fig. 9. Predicted train and test data of CPU usage

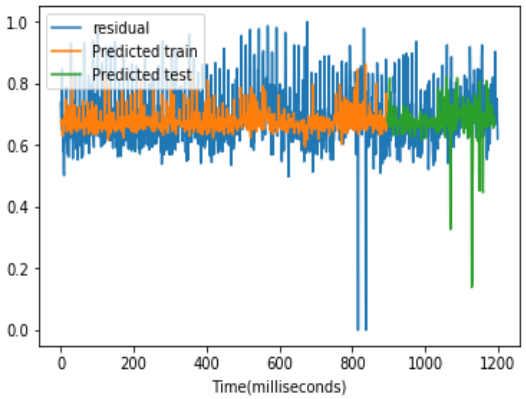


Fig. 10. Predicted train and test data of Memory usage

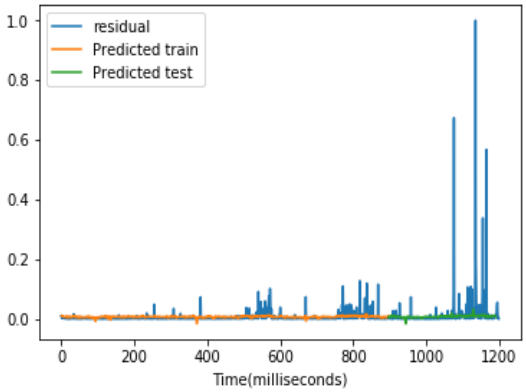


Fig. 11. Predicted train and test data of Disk usage

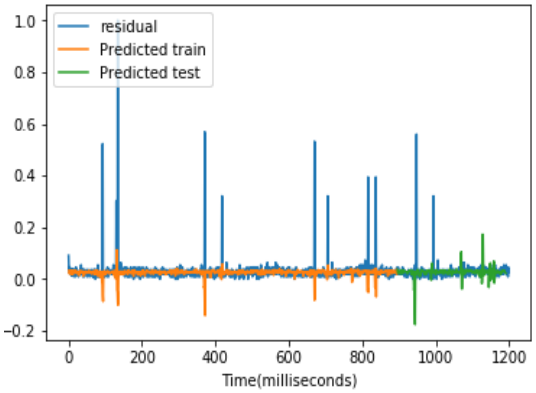


Fig. 12. Predicted train and test data of Network usage

The proposed model indicates lower values of MSE and MAE compared to the ARIMA-LSTM, the VAR-GRU and the VAR-MLP models.

The RMSE value is far lesser in ARIMA-LSTM model. However, the proposed model still additionally shows inferior values of RMSE compared to the remaining models (Table 2).

Table 2. CNN-LSTM model performance results of test data

	The proposed model	ARIMA-LSTM (CPU prediction)	VAR-GRU	VAR-MLP
MSE	0.004798	0.008502	0.005156	0.013245
MAE	0.031813	0.043959	0.035053	0.040430
RMSE	0.069271	0.066552	0.071809	0.115087

4 Conclusions

An important feature of cloud computing is the ability to determine allocated resources based on actual usage. However, this operation requires a start-up time for resource allocation. In order to reduce this time, it is essential to plan in advance the amount of resources needed for the future.

In this paper, we adopted the CNN-LSTM model for multivariate workload prediction in an attempt to extract complex features of the Vm usage components, then model temporal information of irregular trends in the time series components.

The proposed approach was tested using actual data from Bitbrains data. The results are positive and show that the proposed method is more effective than the other predictive models.

References

1. Jiang, Y. et al. 2013. Cloud analytics for capacity planning and instant VM provisioning. *IEEE Transactions on Network and Service Management*. 10, 3 (2013), 312–325. DOI:https://doi.org/10.1109/TNSM.2013.051913.120278
2. S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155-162, 2012.
3. Calheiros, R.N. et al. 2015. Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Transactions on Cloud Computing*. 3, 4 (2015), 449–458. DOI:https://doi.org/10.1109/TCC.2014.2350475.
4. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159–175. doi:10.1016/s0925-2312(01)00702-0
5. Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10), 2451–2471. doi:10.1162/089976600300015015

6. Sims, C. A. (1980). Macroeconomics and Reality. Econometrica, 48(1), 1. doi:10.2307/1912017

Author´s Index

Announcement

World Congress DSA 2021

The Frontiers in Intelligent Data and Signal Analysis July 12 - 23, 2021, New York, USA

www.worldcongressdsa.com

We are inviting you to our fourth World congress on the Frontiers of Signal and Image Analysis DSA 2021 to New York, Germany.

This congress will feature three events:

- the 17th International Conference on Machine Learning and Data Mining MLDM (www.mldm.de),
- the 21th Industrial Conference on Data Mining ICDM (www.data-mining-forum.de),
- and the 16th International Conference on Mass Data Analysis of Signals and Images in Artificial Intelligence&Pattern Recognition MDA-AI&PR (www.mda-signals.de).

Workshops and Tutorial will also be given.

Come to join us to the most exciting event on Intelligent Data and Signal Analysis.

Sincerely your,
Prof. Dr. Petra Perner

MLDM www.mldm.de

icdm www.data-mining-forum.de

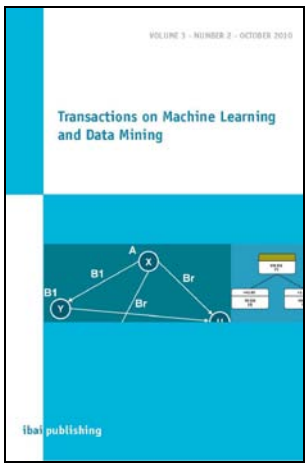
mda www.mda-signals.de

Journals by ibai-publishing

The journals are free on-line journals but having in parallel hardcopies of the journals. The free on-line access to the content of the paper should ensure fast and easy access to new research developments for researchers all over the world. The hardcopy of the journal can be purchased by individuals, companies, and libraries.

Transactions on Machine Learning and Data Mining

P-ISSN: 1865-6781 E-ISSN: 2509-9337



The International Journal "Transactions on Machine Learning and Data Mining" is a periodical appearing twice a year. The journal focuses on novel theoretical work for particular topics in Data Mining and applications on Data Mining.

Net Price (per issue): EURO 100
Germany (per issue): EURO 107 (incl. 7% VAT)

Submission for the journal should be send to:
info@ibai-publishing.org

For more information visited: www.ibai-publishing.org/journal/mldm/about.html

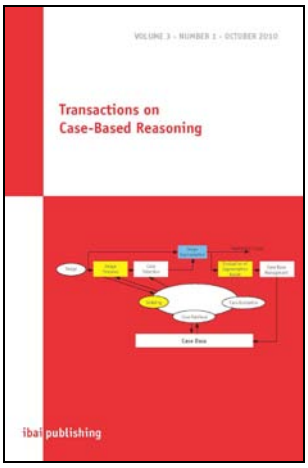
Transactions on Case-Based Reasoning

P-ISSN: 1867-366X E-ISSN: 2509-9345

The International Journal “Transactions on Case-Based Reasoning" is a periodical appearing once a year.

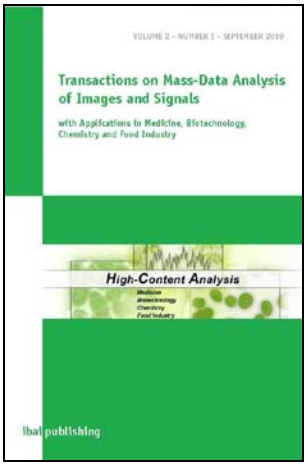
Net Price (per issue): EURO 100
Germany (per issue): EURO 107 (incl. 7% VAT)

Submission for the journal should be send to:
info@ibai-publishing.org



For more information visited: www.ibai-publishing.org/journal/cbr/about.html

Transactions on Mass-Data Analysis of Images and Signals ISSN: 1868-6451 E-ISSN: 2509-9353



The International Journal "Transactions on Mass-Data Analysis of Images and Signals" is a periodical appearing once a year.

The automatic analysis of images and signals in medicine, biotechnology, and chemistry is a challenging and demanding field. Signal-producing procedures by microscopes, spectrometers and other sensors have found their way into wide fields of medicine, biotechnology, economy and environmental analysis. With this arises the problem of the automatic mass analysis of signal information. Signal-interpreting systems which generate automatically the desired target statements from the signals are therefore of compelling necessity. The continuation of mass analyses on the basis of the classical procedures leads to investments of proportions that are not feasible. New procedures and system architectures are therefore required.

Net Price (per issue): EURO 100
Germany (per issue): EURO 107 (incl. 7% VAT)

Submission for the journal should be send to:
info@ibai-publishing.org

For more information visited: www.ibai-publishing.org/journal/massdata/about.php