JYVÄSKYLÄ STUDIES IN COMPUTING

270

Elena Ivannikova

Intelligent Solutions for Real-Life Data-Driven Applications

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 270

Elena Ivannikova

Intelligent Solutions for Real-Life Data-Driven Applications

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Beeta-salissa
joulukuun 14. päivänä 2017 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Beeta hall, on December 14, 2017 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2017

Intelligent Solutions for Real-Life Data-Driven Applications

JYVÄSKYLÄ STUDIES IN COMPUTING 270

Elena Ivannikova

Intelligent Solutions for Real-Life Data-Driven Applications

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2017

Editors
Timo Männikkö
Faculty of Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-7279-0

URN:ISBN:978-951-39-7279-0
ISBN 978-951-39-7279-0 (PDF)

ISBN 978-951-39-7278-3 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2017

ABSTRACT

Ivannikova, Elena
Intelligent solutions for real-life data-driven applications
Jyväskylä: University of Jyväskylä, 2017, 54 p. (+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; **270**)
ISBN 978-951-39-7278-3 (nid.)
ISBN 978-951-39-7279-0 (PDF)
Finnish summary
Diss.

The subject of this thesis belongs to the topic of machine learning or, specifically, to the development of advanced methods for regression analysis, clustering, and anomaly detection. Industry is constantly seeking improved production practices and minimized production time and costs. In connection to this, several industrial case studies are presented in which mathematical models for predicting paper quality were proposed. The most important variables for the prediction models are selected based on information-theoretic measures and regression trees approach.

The rest of the original papers are devoted to unsupervised machine learning. The main focus is developing advanced spectral clustering techniques for community detection and anomaly detection. As part of these efforts, a number of enhancements for the dependence clustering algorithm have been proposed. These enhancements include adding regularization for controlling the size of clusters, extension to the ensemble version for improving model stability, handling overlapping clusters, and adaptation to solving anomaly detection problems and handling big datasets.

Another focus of the thesis is on developing anomaly detection algorithms for network security data. In connection to this, a probabilistic transition-based approach is proposed for detecting application-layer distributed denial-of-service attacks.

The developed approaches are tested on real datasets and are capable of efficiently solving the given tasks with high accuracy and good performance. They are shown to be applicable to solving variable selection, graph segmentation, and anomaly detection tasks in different applications.

Keywords: Clustering, Community detection, Anomaly detection, Paper machine, Regression analysis, Regression trees, Mutual information, Graph segmentation, Spectral clustering, Variable selection, Big data, Network security

Author	Elena Ivannikova Faculty of Information Technology University of Jyväskylä Finland E-mail: elena.v.ivannikova@student.jyu.fi
Supervisor	Professor Dr. Timo Hämäläinen Faculty of Information Technology University of Jyväskylä Finland
Reviewers	Professor Dr. Andrey Garnaev Department of Computer Modelling and Multiprocessor Systems Saint Petersburg State University Russia Professor Dr. Pekka Toivanen School of Computing University of Eastern Finland Finland
Opponent	Docent Dr. Xiao-Zhi Gao Department of Electrical Engineering Aalto University Finland

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor Prof. Dr. Timo Hämäläinen for continues support and guidance. His constant belief in my abilities has encouraged me over the course of my work.

I am grateful to the reviewers of my thesis Prof. Dr. Andrey Garnaev and Prof. Dr. Pekka Toivanen for careful reading of my work and providing constructive feedback.

My sincere thanks go to Dr. Kichun Lee and Dr. Hyunwoo Park for their collaboration and support during writing a journal manuscript. I appreciate very much the assistance and help that Dr. Kichun Lee provided me during my research visit at the Hanyang University (South Korea).

I would like to thank my other collaborators, Dr. Kari Luostarinen, Dr. Mikhail Zolotkhin, Prof. Dr. Gil David, and Dr. Anna Kononova and all other colleagues who have worked with me and supported my research.

I acknowledge the Faculty of Information Technology and its doctoral program, COMAS, for providing funding during my doctoral studies and financial support for conference and research visits. In particular, I want to give thanks to Sami Kollanus who supported me in the early stages of my research. Additional financial support was provided by the HPY Research Foundation and the Nokia Foundation.

I am especially grateful to my husband Dr. Andrey Ivannikov for endless support and encouragement, and for being a continuous source of inspiration for me over the years.

My warmest thanks go to all my friends from all over the world. I am thankful for having you all in my life and highly appreciate all the moments we spent together.

Finally, I want to thank the two my families in Yaroslavl and Kharkov for their support, encouragement, and understanding. Especially, I am thankful to my mom and dad for all their patience, love, help, never-ending support, and for what I am.

ACRONYMS

API	Application Programming Interface
CART	Classification and Regression Trees
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DC	Dependence Clustering
DDoS	Distributed Denial-of-Service
ECG	Electrocardiogram
EGD	Ensemble Group Diffusion
EM	Expectation Maximization
GMM	Gaussian Mixture Model
HDFS	Hadoop Distributed File System
INFLO	Influenced Outlierness
*k***NN**	*k*-Nearest Neighbors
LAD	Least Absolute Deviation
LDA	Latent Dirihlet Allocation
LOF	Local Outlier Factor
LS	Least Squares
MIC	Maximal Information Coefficient
ML	Maximum Likelihood
MLP	Multi-layer Perceptron
NN	Nearest Neighbors
PCA	Principal Component Analysis
SDC	Soft Dependence Clustering
SDN	Software-Defined Network
SOM	Self-Organizing Map
RDD	Resilient Distributed Dataset

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Elena Ivannikova, Timo Hämäläinen, and Kari Luostarinen. Information-theoretic approach to variable selection in predictive models applied to paper machine data. *Proc. of the IEEE Symposium on Computers and Communications, pp. 946-950*, 2013.

PII Elena Ivannikova, Timo Hämäläinen, and Kari Luostarinen. Variable group selection based on regression trees: Paper machine case study. *Proc. of the Conference on Evolving and Adaptive Intelligent Systems, pp. 1-5* , 2014.

PIII Elena Ivannikova, Hyunwoo Park, Timo Hämäläinen, and Kichun Lee. Revealing Community Structures by Ensemble Clustering using Group Diffusion. *Information Fusion, vol. 42, pp. 24-36*, 2018.

PIV Elena Ivannikova, Anna V Kononova, and Timo Hämäläinen. Probabilistic group dependence approach for discovering overlapping clusters. *Proc. of the 26th International Workshop on Machine Learning for Signal Processing, pp. 1-6*, 2016.

PV Elena Ivannikova. Scalable implementation of dependence clustering in Apache Spark. *Proc. of the Conference on Evolving and Adaptive Intelligent Systems, pp. 1-6*, 2017.

PVI Elena Ivannikova, Gil David, and Timo Hämäläinen. Anomaly detection approach to keystroke dynamics based user authentication. *Proc. of the IEEE Symposium on Computers and Communications, pp. 885-889*, 2017.

PVII Elena Ivannikova, Mikhail Zolotukhin, and Timo Hämäläinen . Probabilistic Transition-Based Approach for Detecting Application-Layer DDoS Attacks in Encrypted Software-Defined Networks. *Proc. of the 11th International Conference on Network and System Security, pp. 531-543*, 2017.

1 INTRODUCTION

1.1 Background and research motivation

Currently, the amount and complexity of data being accumulated by companies and requiring analysis is growing exponentially. Data are becoming a major source of generating profit increases for many organizations, with numerous emerging companies' total business activities involving data mining and data analysis. Therefore, business and researchers are focusing on developing efficient algorithms for processing bigger and more diverse datasets.

Data mining involves exploring and analyzing data, seeking patterns or relationships between variables. Based on the given data, a model is built and applied to new data subsets. Therefore, it is of great importance that the model generalizes well. Data mining is a popular toolbox for solving various problems, and this requires revealing data structures to guide decisions when confidence is limited. Data mining tools can be divided into several categories according to the objectives the data analysis aims to achieve. According to [HSM01], these categories include the following

1. *Exploratory Data Analysis*, which refers to exploring the data without specific knowledge of what to extract;
2. *Descriptive Modeling*, which aims to describe all the data or the processes generating the data, such as density estimation, cluster analysis, dependency modeling;
3. *Predictive Modeling*, where a model is built in classification and regression that predicts one variable from the known values of other variables;
4. *Discovering Patterns and Rules*, which focuses on finding useful patterns and rules from data; this is opposed to the three previous categories, which involve model building; and
5. *Retrieval by Content*, applies when data are examined for patterns that are similar to a pattern of interest.

This thesis, which comprises collection of publications, is focused on solving a series of theoretical and practical tasks using data mining techniques, with emphasis on developing clustering and anomaly detection algorithms. Thus, in [PI],[PII] a predictive modeling task is addressed. In these works, industrial data with laboratory measurements collected during test experiments for measuring paper quality are used. The goal is to select a subset of predictor process variables that affect the target quality variable the most. In other words, the aim is to build a model for predicting a target variable from a subset of predictor variables to maximize the prediction accuracy. The rest of the works [PIII]-[PVII] aim at solving descriptive modeling tasks, where advanced clustering and anomaly detection algorithms are proposed for graph segmentation and network security applications.

1.2 Research questions

The goal of the thesis is to develop advanced algorithms for variable selection, community detection, clustering, and anomaly detection, and apply these algorithms for discovering knowledge in datasets from different domains. To meet these goals, the research aims to answer the following research questions:

RQ1: How should variables be selected from data when features dominate over samples to leave prediction accuracy high?

RQ2: How can the community detection/clustering task be made more accurate and efficient?

RQ3: How can anomaly detection techniques be advanced to increase network security?

RQ4: How can community detection/clustering algorithms be efficiently implemented and adapted to large-scale datasets?

1.3 Structure of the thesis

The remaining part of this thesis is organized as follows. Chapter 2 explains the theoretical background. The main concepts and algorithms used in the included publications, as well as those needed for understanding the outcomes of this thesis, are thoroughly described. The main concepts that are addressed in this thesis include supervised learning, unsupervised learning, model selection, and big data analysis. Chapter 3 summarizes the results presented in the included research articles, outlining the research contribution and the author's contribution in the related publications. Finally, Chapter 4 presents conclusions and provides directions for future research work. Table 1 lists the contributions of each included publication with respect to the research questions formulated in Section 1.2.

TABLE 1 Contribution of original research articles to the research questions.

Research Question(s)	Article
RQ1	[PI]: Information-theoretic approach to variable selection in predictive models applied to paper machine data [PII]: Variable group selection based on regression trees: paper machine case study
RQ2	[PIII]: Revealing community structures by ensemble clustering using group diffusion [PIV]: Probabilistic group dependence approach for discovering overlapping clusters [PV]: Scalable implementation of dependence clustering in Apache Spark
RQ3	[PVI]: Anomaly detection approach to keystroke dynamics based user authentication [PVII]: Probabilistic transition-based approach for detecting application-layer DDoS attacks in encrypted software-defined networks
RQ4	[PV]: Scalable implementation of dependence clustering in Apache Spark

2 THEORETICAL FOUNDATIONS

2.1 Supervised learning

In supervised learning, given a set of data $\Omega = \{(\mathbf{x}^{(i)}, y^{(i)}) | i = 1, ..., m; \mathbf{x}^{(i)} \in \mathbb{R}^n; y^{(i)} \in \mathbb{R}\}$, the goal is to learn an approximation $f : \mathbf{x} \rightarrow y$ ($f : \mathbb{R}^n \rightarrow \mathbb{R}$). One usually starts with a family of models $y = f(\mathbf{x}, \Theta)$ parametrized by Θ and aims to find the model parameters $\hat{\Theta}$ that minimize a discrepancy $J(y, \hat{y})$ between the true output $y \in \mathbb{R}$ and expected model output $\hat{y} = f(\mathbf{x}, \hat{\Theta})$. Regression and classification problems belong to supervised learning. Regression methods work with continuous values, while classification methods are used with categorical data. Supervised learning is commonly employed in applications where future events are to be predicted based on information from the past.

2.1.1 Linear regression

Regression analysis is a statistical method that investigates the relationships between a dependent response variable and several independent variables (predictors). Regression analysis is usually used to address the following goals [YS09]:

1. Predicting a response variable y from predictors x_1,...,x_n as accurately as possible;
2. Understanding the structural relationships between the response variable y and predictors x_1,...,x_n; and
3. Determining a subset of variables from predictors x_1,...,x_n that affect the response variable y the most.

Linear regression is a method that assumes linear relationships between input and output variables, where input variables can be original predictors undergoing generally nonlinear transformations. Linear regression with multiple variables, also called multiple linear regression, models the linear relationships between one dependent variable and more than one independent variable. The general form of the multiple linear regression model can be described by the following equation:

$$y(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i + \epsilon,$$

where y denotes the dependent variable, $x_1, ..., x_n$ are n independent variables, $(\alpha_0, ..., \alpha_n)$ define the model parameters, and ϵ is the error term. Classical settings of regression analysis assume the error term ϵ to be normally distributed, with $\mathrm{E}(\epsilon) = 0$ and constant $\mathrm{Var}(\epsilon) = \sigma^2$.

There are different estimators of the regression parameters, for example, least squares estimation or maximum likelihood (ML) [YS09], [HTF09]. The least squares estimation method is considered the most common. In general, given some estimates for the model parameters $\hat{\alpha_i}$, $i = 1, ..., n$ one can predict the response variable using

$$\hat{y} = \hat{\alpha_0} + \sum_{i=1}^{n} \hat{\alpha_i} x_i,$$

where $\hat{y}$ denotes a prediction of $y(\mathbf{x})$.

Least Squares Estimation

Given a multiple linear regression model $y(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$, the least squares cost function to be minimized is written in the following form:

$$J(\alpha) = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2,$$

where $\alpha = (\alpha_0, ..., \alpha_n)$, and $\hat{y}^{(i)}$ is the i-th expected result. Further, using a matrix notation the problem can be represented as

$$y(\mathbf{x}) = [\alpha_0 \, \alpha_1 \cdots \alpha_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \alpha^T \mathbf{x},$$

where $x_0 = 1$. Therefore, given $X \in \mathbb{R}^{m \times (n+1)}$, with each row corresponding to the i-th sample, the cost function can be represented in the matrix form as

$$J(\alpha) = (X\alpha - \hat{y})^T (X\alpha - \hat{y}) = \alpha^T X^T X \alpha - 2(X\alpha)^T \hat{y} + \hat{y}^T \hat{y}. \tag{1}$$

The least squares method aims to find the estimates $\hat{\alpha}$ that minimize the least-squares cost function (1):

$$\hat{\alpha} = \arg\min_{\alpha} J(\alpha).$$

By solving the partial differential system

$$\frac{\partial J}{\partial \alpha} = 0,$$

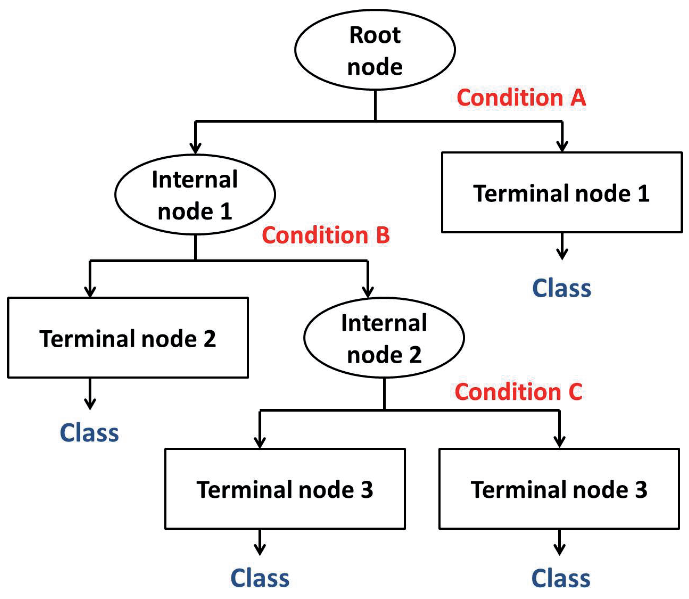


FIGURE 1 Example of classification and regression trees (CART).

it is easy to show that the optimal parameter values in the sense of least squares minimization are equal to

$$\hat{\alpha} = (X^T X)^{(-1)} X^T \hat{y}.$$

It is important to note that linear regression models are not bounded to the analysis of only the linear relationships between the predictor and target variables. The linear property is assumed for the model parameters α_i; that is, the dependence of y on $x_i \leftarrow g(x_i)$ is linear, where g is generally a nonlinear transformation. Hence, nonlinear relationships can be represented naturally using the linear regression model by imposing a linear dependence constraint between y and the transformed predictor space.

2.1.2 Regression trees

Classification and regression trees (CART) are an example of regression models used for predicting continuous variables (regression) or categorical variables (classification) [BFOS84]. A CART tree is a binary decision tree that is constructed by recursively splitting a node into two child nodes, starting with the root node that contains the whole learning sample (see Figure 1 for visual example). CART recursive partitioning is a binary procedure that aims to maximize the similarity of the response variable in each node by utilizing information contained in a set of predictors. Tree construction in CART consists of three major steps, as follows: (1) the process of growing a tree, (2) a splitting criterion, and (3) a validation (stopping) rule for determining the best tree size.

Growing a tree is the process of choosing a split among all the possible splits at each node so that the resulting child nodes are the "purest", that is, the distribution of the outcomes of all points belonging to each node has the lowest variability. The growing process starts from the root node by recursively using the following steps on each node:

1: Find each predictor's best split, the point that maximizes the splitting criterion the most when the node is split according to it;
2: Among the best splits found in the previous step, choose the one that maximizes the splitting criterion, the node's best split; and
3: If a stopping criterion is fulfilled, exit. Otherwise, apply step 1 to each child node in turn.

The goodness of split is measured by an impurity function. There are two impurity functions that are used in regression trees: (1) the least squares (LS) function and (2) the least absolute deviation (LAD) function [YW99]. Here, only the LS measure is discussed, since the mechanism for both procedures is the same. Given a splitting variable $j \in \{1, ..., n\}$ and a split point $s \in \mathbb{R}$, the node impurity under the LS criterion is measured by the within-node sum of squares $S(j)$, defined as

$$S(j) = \sum_{i \in R} (y_{i(j)} - \bar{y}_{(j)})^2,$$

where $\bar{y}_{(j)}$ denotes the mean value of the response variable at node R. The best split can be found by solving the following optimization problem:

$$\max_{(j,s)} J(j,s) = S(j) - S(R_1(j,s)) - S(R_2(j,s)),$$

where $R_1(j,s) = \{x_i, i \in R | x_{ij} \leq s\}$ and $R_2(j,s) = \{x_i, i \in R | x_{ij} > s\}$. Therefore, $S(R_1(j,s))$ is the sum of squares of the left R's child node, and $S(R_2(j,s))$ is the sum of squares of the right R's child node. In other words, the splitting rule serves to choose the split that causes the maximum reduction in the impurity of the parent node.

To prevent the tree from growing too large, and thus, overfitting, one should decide the optimal size of a tree. The standard approach is to grow a "full" tree T_{max} and then perform pruning by minimizing the cost-complexity measure on the cross-validation set. The cost-complexity measure can be written in the form

$$R_\alpha(T) = R(T) + \alpha|T|,$$

where $T \in T_{max}$ is any subtree obtained by pruning, $|T|$ determines the number of terminal nodes in T, $\alpha \geq 0$ is the complexity parameter, and $R(T)$ refers to the sum of misclassification errors (sum of square errors) over all nodes. The goal is to find, given α, the subtree T_α that minimizes $R_\alpha(T)$:

$$T_\alpha = \arg\min_{T \in T_{max}} R_\alpha(T).$$

The parameter α affects the size of the subtree to be selected. Thus, when $\alpha = 0$, the biggest tree will be chosen, and with an increase in α the tree size becomes smaller.

CART are easy to interpret and can handle mixed (discrete and continuous) and missing data; moreover, they are resistant to outliers and monotone transformations in data. CART have been used in many applications, including health systems, business, ecology, and network planning [FJBD06], [LKO11], [DF10], [HK07]. In [PII], regression trees are utilized to discover the best pairs of predictor variables. Despite their advantages, however, the trees can be unstable, causing a decrease in prediction accuracy. Due to the hierarchical nature of the tree-growing process, small changes to the input data can have large effects on the structure of the tree [Mur12]. As discussed in Section 2.3.2, one way to reduce the variance of an estimate is bagging. *Random forests* [Bre01] improve the accuracy of regression trees by averaging multiple individual trees that are trained on a randomly chosen subset of input variables, along with a randomly chosen subset of data samples. Random forests are applicable in many areas, including image classification, geography, ecology, and bioinformatics [GBS06], [BZM07], [CEB+07], [SWA08].

2.1.3 *k*-Nearest Neighbor

k-Nearest Neighbors (*k*NN) is a simple classifier that selects the *k* nearest points to a test input in the training set and classifies the test point using the majority vote among the *k* neighbors. Let us denote the training subset by $\mathrm{Y} = \{y_i\}, i = 1, ..., N$, the test input by $\mathbf{x}$, the indices of the k nearest points to $\mathbf{x}$ in Y by $\mathrm{I}_k(\mathbf{x}, \mathrm{Y})$, and the set of classes by $c_j, j = 1, .., C$. The class is defined as follows:

$$\arg\max_{j} \frac{1}{k} \sum_{i \in \mathrm{I}_k(\mathrm{x},\mathrm{Y})} \mathbb{I}(y_i = c_j),$$

where $\mathbb{I}(\alpha) = 1$ if α is true and 0 otherwise.

The *k*NN classifier can work well when it is given a good distance measure and has enough labeled training data, but it fails in high dimensions due to the *curse of dimensionality* [HTF09]. In *k*NN, any distance measure can be used, but the most common is Euclidean distance. *k*NN has been successfully applied to many classification problems, including handwritten digits, satellite image scenes, and electrocardiogram (ECG) signals [ZBA12], [STB00], [Lee91], [MNW02], [CMP03], [SSK13], [KAKN06]. In [PVI], a *k*NN based approach was used for detecting anomalous users during the authentication phase.

2.1.4 Multilayer Perceptron

Artificial neural networks [MP43] were originally designed as algorithms attempting to mimic human brain. They represent an effective state-of-the-art technique for many machine learning applications. Neural networks comprise artificial neurons, information-processing units that are fundamental to the operation of the neural network [Hay11]. A simple mathematical model for the activity of one artificial neuron can be written as follows:

$$y = \phi(a_0 + \sum_{i=1}^{m} \omega_i x_i), \tag{2}$$

where y is the output of the neuron, $x_i, i = 1, ..., m$ are the inputs connected to the neuron, ω_i are the corresponding synaptic weights of the neuron, ϕ is an activation function, and a_0 is bias. The equation 3 is often written in the form

$$y = \phi(\sum_{i=0}^{m} \omega_i x_i), \tag{3}$$

where $\omega_0 = a_0$ and $x_0 = 1$.

The activation function ϕ defines the output of a neuron and can be *linear*, *threshold* (the output depends on a threshold value), or *sigmoid*, where $g(x) = (1 + e^{-x})^{-1}$, which is one of the most widely used activation functions. Other popular activation functions are the *tanh* function $f(x) = tanh(x) = 2/(1 + e^{-2x})^{-1} - 1 = 2 \cdot sigmoid(2x) - 1$ and a more recent *ReLu* function $h(x) = \max(0, x)$, which is less computationally expensive than *sigmoid* and *tanh* due to involving simpler mathematical operations. The introduction of *ReLu* was one of the keys to success of modern neural networks, as it allows faster training and leads to sparse activations. Since, on the right end the derivative does not tend to zero compared with *sigmoid* or *tanh*, it helps to tackle the problem of vanishing gradients [Hay11].

Neural networks can be represented as graphs with connected layers of nodes, where each node represents a neuron. Each neuron computes a sum of its inputs (activation from previous layers) weighted by the synaptic weights and applies the activation function, resulting in activation that is propagated to the next layer.

The way in which neurons are connected to each other defines the network architecture, or network topology. For example, *feed-forward networks* allow connections to be only one way, from input to output. *Multilayer perceptron* (MLP) belongs to the class of feed-forward networks. The models that allow feedback connections are called *recurrent neural networks*. They are powerful and dynamic, and they can become extremely complicated. In *Hopfield network* (*associative memory*) models, symmetric connections between the hidden units are allowed [Mur12].

An MLP is represented as a series of logistic regression models. Depending on whether a classification or regression problem is being solved, the final layer can either be another logistic regression or a linear regression model. There is an input layer, output layer, and one or more hidden layers. The activity of a node is determined by the activities of the previous nodes and their interconnections' weights. Thus, every node is assigned an input-output (activation) function that, along with the weights, influences the behavior of the network. An example of an MLP is shown in Figure 2. In this dissertation, the MLP was used for predicting paper quality from laboratory measurements, as described in [PI].

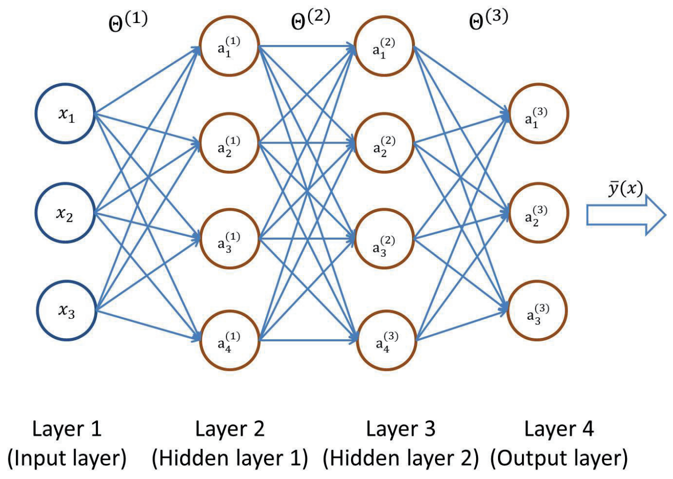


FIGURE 2 Example of a multilayer perceptron (MLP).

The back-propagation algorithm

The model parameters should be adjusted to make the model fit the training data well. A solution for the learning problem is the weight combination that minimizes the error function. Using gradient descent, the back-propagation algorithm seeks the minimum of the error function in the weight space.

Given a training set $\{(\mathbf{x}^{(i)},y^{(i)})\}$, $i = 1, ..., m$, with the parameter vector $\Theta \in \mathbb{R}^n$, let us denote a total number of layers in the network as L and the number of units (neurons) in layer l as s_l. The number of output units equals the number of classes K one wants to predict. Let us denote the activation function as g, the output as $h_\Theta(\mathbf{x}) \in \mathbb{R}^K$, and the hypothesis that results in the k-th output as $h_\Theta(\mathbf{x})_k$. The error function that is usually used in regression is the sum of squared errors:

$$J(\Theta) = \sum_{i=1}^{m}\sum_{k=1}^{K}(y_k^{(i)} - h_\Theta(\mathbf{x}^{(i)})_k)^2 - \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{p=1}^{s_l}\sum_{q=1}^{s_{l+1}}(\Theta_{qp}^{(l)})^2. \tag{4}$$

For multiclass classification, the error function is usually given in the form of cross entropy:

$$J(\Theta) = -\sum_{i=1}^{m}\sum_{k=1}^{K}(y_k^{(i)}\log(h_\Theta(\mathbf{x}^{(i)}))_k - \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{p=1}^{s_l}\sum_{q=1}^{s_{l+1}}(\Theta_{qp}^{(l)})^2. \tag{5}$$

Neural networks can be applied to a problem of multilabel classification. For the problem of multilabel classification, the most common choice of error function is a generalized form of logistic regression:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(\mathbf{x}^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(\mathbf{x}^{(i)}))_k\right]$$
$$- \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{p=1}^{s_l}\sum_{q=1}^{s_{l+1}}(\Theta_{qp}^{(l)})^2. \tag{6}$$

In (6), $\Theta^{(l)}$ is a matrix with the number of columns equal to the number of nodes in the layer l (including the bias unit) and the number of rows equal to the number of nodes in the next layer $(l+1)$ (excluding the bias unit).

The gradient descent method involves several steps. First, forward propagation is applied to compute the hypothesis $h_\Theta(\mathbf{x})$. Then, the back-propagation algorithm is applied to compute the derivatives $\frac{\partial}{\partial \Theta_{qp}^{(l)}} J(\Theta)$. Let us denote the error of node p in layer l by $\delta_p^{(l)}$. The gradient descent method for the error function (6) can be described as follows:

```
1: Set $\Delta_{qp}^{(l)} = 0, \forall\, l,p,q$.
2: **for** $i = 1$ to $m$ **do**
3:     $a^{(1)} = \mathbf{x}^{(i)}$.
4:     Compute $a^{(l)}$ for $l = 2, ..., L$ using forward propagation;
       $a^{(l)} = g(\Theta^{(l-1)} a^{(l-1)})$.
5:     Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$.
6:     Compute $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)} \circ a^{(l)} \circ (1 - a^{(l)})$ for $l = L-1, ..., 2$.
7:     $\Delta_{(qp)}^{(l)} := \Delta_{(qp)}^{(l)} + a_p^{(l)} \delta_q^{(l+1)}$.
8: **end for**
9:
   $D_{ij}^{(l)} := \begin{cases} \frac{1}{m}\Delta_{qp}^{(l)} + \lambda\Theta_{qp}^{(l)} & \text{if } p \neq 0, \\ \frac{1}{m}\Delta_{qp}^{(l)} & \text{if } p = 0. \end{cases}$
10: $\frac{\partial}{\partial \Theta_{qp}^{(l)}} J(\Theta) = D_{qp}^{(l)}$.
11: $\Theta_{qp}^{(l)} \leftarrow \Theta_{qp}^{(l)} - \alpha D_{qp}^{(l)}$.
```

Here, $\circ$ denotes element-wise multiplication also known as the Hadamard product, and α denotes the learning rate.

2.1.5 Regularization

Every model can overfit. Overfitting happens when a model appears to be too powerful for the training data, and as a result, it can memorize data, including noise. Regularization is a technique used to prevent overfitting during the training process. *Early stopping* is a simple way to prevent overfitting. This method states that as soon as the error on the validation set starts to increase, the training procedure should stop. If it continues to train, then the initially simple model becomes more complicated as training progresses, and it eventually overfits.

Another classical approach to preventing overfitting is L_1 and L_2 regularization [Mur12]. This manifests in adding a penalty term for parameters to the cost function (e.g., as in (4)-(6)). In a probabilistic view, this is equivalent to imposing a prior on the parameters. The L_1 norm usually leads to sparser solutions for parameters compared with the L_2 norm.

Dropout [SHK$^+$14] is a relatively recent technique for regularization in neural networks that has especially influenced tremendous success of deep learning. It decreases overfitting and improves a training process's speed. Another effect of dropout is that it prevents co-adaptations of neurons and forces them to specialize on certain tasks. In dropout, at each training iteration, a portion of the activations in each layer is randomly masked out according to a predefined probability. This effectively results in training an ensemble of reduced networks. After the training finishes, the entire network is used when applied to test examples, and one must re-normalize the network weights according to the dropout probability to match this.

Batch normalization [IS15] is another popular and widely accepted regularization technique that is commonly used in deep neural networks [GBC16]. In batch normalization, not only the inputs to the network, but also the inputs to every layer are normalized. The normalization is made a part of the model architecture to prevent stochastic gradient descent [Hay98] from undoing the effect of normalization. Batch normalization can often result in order-of-magnitude training speedups, and it tends to reduce overfitting. Batch normalization has the effect of reducing the internal covariate shift, thereby making the input distribution of a layer more stable.

2.2 Unsupervised learning

In contrast to supervised learning, which is usually used for prediction and regression, the goal of unsupervised learning is to discover interesting structures (knowledge) in data. In unsupervised learning, the labels are unknown beforehand. Moreover, labeling data is an expensive process that normally involves human experts, and well-labeled data are rarely available. On a high level, the goal of the unsupervised learning can be to learn a probability distribution $p(\mathbf{X})$ over the observed data or a generative model of the data $\Omega = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)} | \mathbf{x}^{(i)} \in \mathbb{R}^n\}$. If only a kernel (pairwise similarities) [HSS08] is given, then one can still search for clusters of data points.

2.2.1 Clustering

Clustering is a process of partitioning unlabeled data points into homogeneous groups. The data points are grouped in such a way that they are similar to one another in each cluster and different from points in other clusters [HTF09]. Clustering is also referred to as *community detection* or *graph partitioning*, and it has many applications in data mining, machine learning, signal and image processing, network analysis, pattern recognition, bioinformatics, and so on.

Similarity measures

The definition of similarity between objects to be clustered is fundamental in cluster analysis. A measure of similarity between two objects in a feature space is essential to most clustering algorithms. Depending on the similarity measure, the outcomes of the algorithms can differ significantly [JMF99]. The most common way to calculate dissimilarity between the objects is using a distance measure. Due to the variety of data types and scales, the distance measure must be chosen carefully.

For continuous features, the most common choice of metric is *Euclidean distance*. Given $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^n$ the Euclidean distance, calculated as

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = ||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2 = \sqrt{\sum_{k=1}^{n} (x_k^{(i)} - x_k^{(j)})^2}, \tag{7}$$

works well for data with an underlying clustering structure when the groups are compact or well separated [JMF99]. To account for correlations in the data a scale-invariant *Mahalanobis distance* can be used:

$$d_M(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sqrt{(\mathbf{x}^{(i)} - \mathbf{x}^{(j)}) S^{-1} (\mathbf{x}^{(i)} - \mathbf{x}^{(j)})^T}, \tag{8}$$

where S is the covariance matrix of a random vector variable $\mathbf{X} \in \mathbb{R}^n$ assuming $\mathbf{x}^{(i)} \sim \mathbf{X}$ and $\mathbf{x}^{(j)} \sim \mathbf{X}$. When all axes are rescaled to have unit variance, Mahalanobis distance resembles Euclidean distance in the transformed space.

The calculation of the distance between data instances is more challenging when a dataset contains features of different types, that is, continuous and nominal features, since the contribution of features should be fairly balanced. A distance measure that incorporates both continuous and nominal features is known as a heterogeneous distance measure. Heterogeneous distances can be computed after performing the standardization of features or through a linear combination of distance matrices computed for each set of homogeneous features. Some proximity measures have been introduced for calculating distances between the features of heterogeneous types [WM97], [SPB10]. Sometimes, for example, in social networks analysis, the data are represented in terms of the pairwise proximity (similarity or dissimilarity) between the objects. Thus, some algorithms that assume distances cannot be used with such data.

The main drawback of clustering algorithms is the lack of standardization, which means that depending on the type of dataset, the same algorithms can provide good or poor results. So far, there has not been a standardized method developed that would work well for all practical problems.

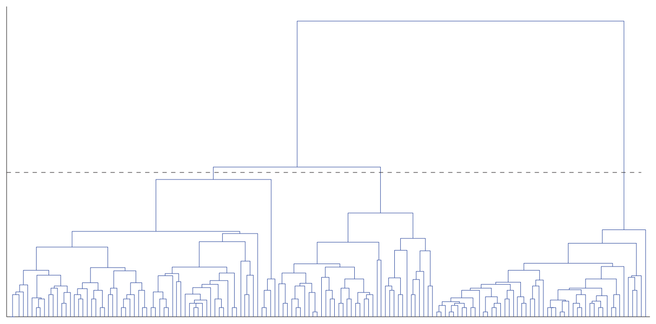


FIGURE 3 Example of a hierarchical cluster tree for Fisher's dataset.

Types of clustering algorithms

Among the most popular clustering approaches are hierarchical (agglomerative or divisive) clustering, centroid-based clustering, density-based clustering, spectral clustering, and probabilistic clustering.

Hierarchical clustering defines a similarity measure between clusters and computes a similarity matrix between vertices of a graph. Agglomerative clustering algorithms start with assigning each point to its own cluster and then recursively merging a selected pair of clusters into a single cluster. A pair of clusters to be merged is selected to minimize intergroup dissimilarity. In contrast, divisive clustering methods start by assigning all points to a single cluster and then recursively splitting one of the existing clusters into two new clusters. The splitting is performed to maximize between-group dissimilarity. Figure 3 shows an example of a hierarchical cluster tree for Fisher's dataset [Fis36]. One of the possible divisions into three clusters is shown by the dashed line.

Spectral clustering refers to the group of methods based on eigenvalue decomposition of the similarity matrix or its derivative matrices for clustering datasets. This approach can take into account geometric structures of the data, and it does not make any assumptions about the shape of clusters; thus, it is good at finding non-convex clusters.

Density-based methods group data points according to regions of density. Thus, the points from disconnected regions of high density are assigned to different clusters, while the rest are marked as noise.

In *centroid-based* (prototype) methods, each class is represented by its centroid (or medoid for categorical data attributes), the most central point. The data points are assigned to a cluster with the nearest centroid.

Probabilistic clustering approaches consider data to be a sample independently drawn from a mixture model of several probability distributions [MB88]. Such mixture models can naturally be generalized to clustering heterogeneous data.

2.2.1.1 Gaussian mixture model

A Gaussian mixture model (GMM) is a probabilistic model that assumes that data are generated from a mixture of a finite number of Gaussian distributions, each representing a different cluster, with unknown parameters. By using a discrete set of Gaussian functions, each with its own mean and covariance matrix, the GMM performs better modeling and can smoothly approximate even arbitrarily shaped densities (clusters).

Let Ω denote a random sample of size m with density function $f(\mathbf{x})$. In GMM, each base distribution is a mixture of N parametric multivariate Gaussian densities in the following form:

$$g(\mathbf{x}|\Theta) = \sum_{k=1}^{N} \omega_i f(\mathbf{x}|\mu_k, \Sigma_k),$$

where $\omega_k \in [0,1], k = 1, .., N$ are the weights of the k-th components of the mixture, under the constraint that $\sum_{k=1}^{N} \omega_k = 1, \Theta = \{\omega_1, ..., \omega_{N-1}, \mu_1, ..., \mu_N, \Sigma_1, ..., \Sigma_N\}$ denotes the parameters of the mixture model, and $f(\mathbf{x}|\mu_k, \Sigma_k)$ are the k-th component Gaussian densities with mean vector μ_k and covariance matrix Σ_k. Multivariate Gaussian distribution is given in the following form:

$$f(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\omega|\Sigma|)^{1/2}} \exp\Big\{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\Big\}. \tag{9}$$

To estimate the model parameters Θ in the GMM, a common approach is to apply the *expectation maximization* (EM) algorithm [DLR77]. EM is an iterative method for numerically approximating the ML estimates of the parameters in a mixture model. The EM algorithm consists of two phases - the initial expectation (E)-step, which estimates the components the data points belong to, and the following maximization (M)-step, which re-estimates the parameters based on the estimation from the previous step.

Assume that the number of mixed components N is known. The log-likelihood is given by the following function.

$$l(\Theta) = \log g(\Omega|\Theta) = \sum_{i=1}^{m} \log g(\mathbf{x}^{(i)}|\Theta) = \sum_{i=1}^{m} \log \Big(\sum_{k=1}^{N} \omega_k f(\mathbf{x}_i|\mu_k, \Sigma_k)\Big). \tag{10}$$

The EM algorithm can be described by the steps given below.

```
Require: Initial guesses for the parameters $\Theta^{(0)}$
1: Evaluate initial value of $l(\Theta^{(0)})$ using (10), $\hat{\Theta} = \Theta^{(0)}$
2: E-step. Using current parameters estimate the responsibilities
```

$$p_k(\mathbf{x}^{(i)}) = \frac{\hat{\omega}_k f(\mathbf{x}^{(i)}|\hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{j=1}^{N} \hat{\omega}_j f(\mathbf{x}^{(i)}|\hat{\mu}_j, \hat{\Sigma}_j)}.$$

3: *M-step*. Using the current responsibilities re-estimate the parameters

$$\hat{\mu}_k = \frac{\sum_{i=1}^m p_k(\mathbf{x}^{(i)})\mathbf{x}^{(i)}}{\sum_{i=1}^m p_k(\mathbf{x}^{(i)})},$$
$$\hat{\Sigma}_k = \frac{\sum_{i=1}^m p_k(\mathbf{x}^{(i)})(\mathbf{x}^{(i)} - \mu_k)(\mathbf{x}^{(i)} - \mu_k)^T}{\sum_{i=1}^m p_k(\mathbf{x}^{(i)})},$$
$$\hat{\omega}_k = \frac{1}{m}\sum_{i=1}^m p_k(\mathbf{x}^{(i)}).$$

4: Evaluate $l(\Theta)$. If there is no convergence, return to step 2.

Therefore, given Ω, the clustering procedure aims to obtain the corresponding labels $\mathbf{y}^{(i)} \in \mathbb{R}^N, i = 1, ..., m$ by estimating the posterior probabilities of component membership for a fitted GMM $p_k(\mathbf{x}^{(i)}) = Pr(y_k^{(i)} = 1|\mathbf{x}^{(i)})$. The weights of Gaussian components can be viewed as prior probabilities of assigning a data point to a given component k, as $\omega_k = Pr(y_k^{(i)} = 1)$. Further, each data point can be assigned to a specific component/cluster according to the maximum a posteriori (MAP) estimation rule when

$$\begin{cases} y_k^{(i)} = 1, & \text{if } k = \arg\max_j p_j \mathbf{x}^{(i)}, \\ 0, & \text{otherwise.} \end{cases}$$

2.2.1.2 *k*-Means clustering

k-Means [Mac67], [HTF09], [Mur12] is a specific case of EM algorithm for a GMM. In contrast to the GMM, which generates soft clustering assignments, *k*-means is a hard clustering algorithm that deals with divisions into non-overlapping groups. The problem of parameter estimation is reduced to estimating the cluster centroids. Given a sample Ω and number of clusters k, one randomly specifies the initial k cluster centroids $\mu_j, j = 1, ..., k$. Then, until the algorithm is converged, the following two steps are performed:

1. Assigning each observation to the closest cluster centroid:
   $c_i = \arg\min_{1 \leq j < k} \|\mathbf{x}^{(i)} - \mu_j\|^2$; and
2. Updating cluster centroids: $\mu_j = (\sum_{i=1}^m I(c_i = j))^{-1} \sum_{i:c_i=j} \mathbf{x}^{(i)}$.

2.2.1.3 Spectral clustering

Spectral clustering [HTF09] refers to the group of methods based on the eigenvalue decomposition of the similarity matrix or its derivative matrices for clustering datasets. This approach can take into account geometric structures of the data, and therefore, it is good at finding non-convex clusters. Compared to the traditional algorithms, such as *k*-means or single linkage, spectral clustering has many fundamental advantages. The main idea behind spectral clustering lies in integrating local neighborhood connections to find strongly connected data components. Spectral clustering is easy to implement and can be solved efficiently using standard linear algebra methods. There are different interpretations of spectral clustering [Lux07]. Here, the graph cut and random walk-based points of view are discussed.

The fundamental concept of spectral clustering algorithms is the graph Laplacian [SM00], [NJW01], [Lux07]. The algorithms and their outcomes vary depending on what form of graph Laplacian has been used. Given a set of data points/observations $\Omega = \{\mathbf{x}^{(i)} | i = 1, ..., m; \mathbf{x}^{(i)} \in \mathbb{R}^n\}$, one can represent observations in the form of a graph defined by an $m \times m$ similarity/affinity matrix S. The similarity, or adjacency, matrix S, symmetric by definition, consists of non-negative elements s_{ij} representing pairwise similarities between points i and j. The closeness between points i and j is usually reciprocal to the distance between i and j. The bigger the value of s_{ij} is, the closer the two points i and j are to each other. Let G be a diagonal matrix with diagonal elements

$$g_i = \sum_{j \in \Omega} s_{ij} \tag{11}$$

which are calculated as the sum of the weights of the edges connected to them and are called the degree of point i. The unnormalized graph Laplacian matrix is defined as

$$\mathrm{L} = \mathrm{G} - \mathrm{S}.$$

In spectral clustering, one finds the k eigenvectors corresponding to the k smallest eigenvalues of the graph Laplacian L. The rows of the obtained eigenvectors are then clustered using a standard clustering technique, such as k-means, to provide a clustering of the original data points. An example of spectral clustering algorithm can be described using the general steps outlined below.

```
Require: Similarity matrix $\mathrm{S} \in \mathbb{R}^{m \times m}$, parameter $k$ specifying number of clusters.
1: Construct similarity graph $G$.
2: Calculate Laplacian L.
3: Compute the first $k$ eigenvectors $\mathbf{e}_1, ..., \mathbf{e}_k$ of L .
4: Form the matrix $\mathrm{E} = [\mathbf{e}_1, ..., \mathbf{e}_k] \in \mathbb{R}^{m \times k}$ such that the eigenvectors are placed
   in columns.
5: Cluster rows of E as points in $\mathbb{R}^k$ using $k$-means algorithm.
6: Assign an original data point to cluster $i$ if the corresponding row of matrix E
   was assigned to cluster $i$.
```

The graph cut interpretation

From the graph cut point of view the observations are represented as an undirected similarity graph $G =< V, E >$, where V and E define vertices and weighted edges, respectively. Vertices $v_i \in V$ represent observations $\mathbf{x}^{(i)}$. A pair of vertices is connected by an edge if the similarity s_{ij} between the corresponding data points is positive and the edge is assigned the weight s_{ij}. Spectral clustering can be interpreted as finding partitions of the graph such that within-group edges have high weights and between-group edges have significantly lower weights [Lux07]. Thus, the algorithm aims to construct similarity graphs that represent the local neighborhood relationships between observations, and the problem of clustering is reduced to a graph-partition problem, where strongly connected components are identified as clusters.

There are different ways to define the similarity measure when constructing an affinity matrix/similarity graph; these are usually application-driven and chosen to better reflect the local behavior and nature of the data and problem. To give some examples, the Gaussian radial basis function and mutual *k*NN graph [PS85] are considered the most common choices.

One way to find a partition into k clusters $C_1, ..., C_k$ is by solving the mincut problem, that is choosing the $C_1, ..., C_k$ that minimizes

$$\mathrm{cut}(C_1, ..., C_k) := \sum_{i=1}^{k} \mathrm{cut}(C_i, \bar{C}_i),$$

where $\bar{C}_i = V \setminus C_i$ is a complement of the subset C_i, $\mathrm{cut}(C_i, \bar{C}_i) = \sum_{p \in C_i, q \in \bar{C}_i} s_{pq}$. However, often, the mincut solution involves cutting a single vertex from the rest of the graph. To avoid division into too small clusters, either the RatioCut [HK92] or normalized Ncut [SM00] objective function can be optimized; these differ in how they measure the size of a subset of a graph

$$\mathrm{RatioCut}(C_1, ..., C_k) := \sum_{i=1}^{k} \frac{\mathrm{cut}(C_i, \bar{C}_i)}{|C_i|},$$
$$\mathrm{NCut}(C_1, ..., C_k) := \sum_{i=1}^{k} \frac{\mathrm{cut}(C_i, \bar{C}_i)}{\mathrm{vol}(C_i)},$$

where $|C_i|$ denotes the number of vertices, and $\mathrm{vol}(C_i) = \sum_{j \in C_i} g_j$, where g_j is the degree of node j as defined in (11).

The probabilistic interpretation

From the probabilistic point of view, the pairwise similarities can be viewed as edge flows in a Markov random walk [MS01], [Lux07]. A Markov random walk on a graph is a stochastic process that randomly transitions among vertices and satisfies the Markov property, wherein the predictions for the future of the process are based solely on its present state. Spectral clustering can be interpreted as trying to find a partition of the graph such that the random walk stays in the same cluster for a long time and rarely jumps between clusters. Thus, the clustering is performed by studying the properties of the eigenvectors and the eigenvalues of the resulting transition matrix, defined as

$$\mathrm{P} = \mathrm{D}^{-1}\mathrm{S}. \tag{12}$$

In (12), P is a stochastic matrix whose rows sum up to one, and elements p_{ij} represent the probability of moving from node i to node j in one step, provided the current location is node i. The first eigenvector of the matrix P is the vector with all elements equal to one. Therefore, the second largest eigenvector of P can be used to describe the cluster properties of the graph.

To conclude, the main advantages of spectral clustering are its efficiency, and the lack of assumptions concerning the clusters' shape. Spectral clustering does not require clusters to be convex, and it can be used for solving general problems. Given a sparse similarity graph, the spectral clustering algorithm can even be efficiently implemented for large datasets. Its main disadvantage is instability with respect to the choice of parameters. Specifically, the algorithm's stability depends on the choices of the parameters for the neighborhood graphs.

2.2.2 Community detection

Community detection algorithms are applied in complex systems/networks represented as graphs. Such algorithms attempt to separate vertices into clusters such that there are many connecting edges among the vertices of the same cluster and relatively few edges joining vertices from different clusters. Community detection plays an important role in physics, sociology [SC11], biology [DWM02], [JTA$^+$00], and computer science, where systems are often represented by graphs, and it has become a central concept in understanding the functionality of complex networks.

A simple example of a graph is a random graph where all the edges have the same weight and the distribution of the edges among the vertices is homogeneous. In real systems, however, the graph structure is more complex, since the distribution of the edges is inhomogeneous not only globally, but also locally. In this case, a network is said to display a *community structure* if the vertices of the network can be easily grouped into communities/clusters such that the edge density is high inside the clusters and low between the clusters.

Identifying network communities is one of the most important tasks in network analysis. The structure of a network is the only preliminary information available to an algorithm. This information includes connections between pairs of vertices and possibly their weights. In real systems, vertices can belong to one community or multiple communities at once. The research related to identifying the community structure includes both overlapping and non-overlapping community detection techniques. The task of overlapping community detection is more recent compared with revealing non-overlapping structures.

Generally, the community detection task is similar to clustering. The main difference is that data-clustering algorithms aim to find groups of objects with similar properties/attributes using a measure of distance or similarity. Communities in graphs are related to the concept of edge density [For10], and community detection algorithms operate on a network structure level. Despite the considerable interest of the scientific community in community detection over the last few years, there is no universally accepted solution so far. Numerous methods have been proposed to reveal the underlying structure in complex networks.

Traditional community detection techniques include hierarchical clustering, partitional clustering, spectral clustering, and graph partitioning [For10]. *Hierarchical* clustering algorithms [HTF09] are commonly applied in networks with a hierarchical structure, such as social networks, where communities have a multilevel structure by nature. Among the most popular *partitional* methods are k-means clustering [Mac67] and fuzzy c-means clustering [Bez81]. *Spectral* methods exploit spectral properties of the graph Laplacian matrices [DM04], [PL14], [PIII], [PIV]. *Graph partitioning* methods [Pot97] divide the vertices of the graph into k groups by minimizing the *cut size* (number of edges lying between clusters). The division is done by iterative bisectioning of the graph [For10]. In graph partitioning, the number of clusters is known beforehand.

Among other approaches are *divisive* methods [GN02] that aim to detect and remove the edges that connect vertices from different communities [For10] based on a vertex betweenness measure [GN02], [Fre77].

Optimization-based community detection techniques involve the optimization of a clustering quality function over the space of all possible partitions. Such quality functions represent the goodness of division. A benchmark quality function is the modularity [NG04], which measures how different the original graph is from its randomizations. Modularity optimization may lead to resolution problems [FB07]. Another approach is optimization of the cluster quality functions, which considers communities as local structures, allows investigating networks by parts, and avoids resolution problems by ignoring the global scale [BGK+05], [HSL+11], [LF11].

2.2.2.1 Overlapping community detection

Overlapping community detection, or soft clustering, techniques can be divided into two classes, namely node-based and link-based approaches. The node-based overlapping community detection algorithms categorize the nodes of the network, while the link-based overlapping community detection algorithms classify the edges of the network in clusters. Having multiple memberships for the nodes increases the space of possible solutions. In overlapping communities, especially in social networks, the links reflect different types of relationships between the objects, and they are usually uniquely defined in contrast to nodes [ABL10]. Thus, multiple links connected to a single node may belong to several different link communities. In link communities, a node is overlapping with other nodes if the links connected to it belong to more than one group. Link clustering [ABL10], clique percolation method [PDFV05], and mixed membership stochastic blockmodels [ABFX08] are considered the state-of-the-art overlapping community detection methods. The soft dependence clustering algorithm proposed in [PIV] can be used for graph segmentation applications, as well as clustering data that have the notion of distance.

2.2.3 Anomaly detection

Anomaly detection refers to a problem of finding patterns in data that significantly deviate from the expected behavior [CBK09]. There are different names for such non-conforming patterns, but the most frequently used terms in the field of anomaly detection are *anomalies* and *outliers*. Anomalies can represent one of several types, depending on their nature and behavior. The simplest anomaly is a *point* anomaly, which refers to a single anomalous sample with regard to the rest of the data. This type of anomaly is the focus of most works on anomaly detection. *Collective* anomaly refers to a group of related samples which is anomalous with respect to the entire dataset. In collective anomaly, not the group samples by themselves, but the presence of the samples as a group is anomalous. Collective anomaly may appear only in data where the data points are related (e.g. graph, sequence, spatial). *Contextual* anomaly is a single point or group of data points that are anomalous with respect to the context in the data [CBK09].

Depending on the labeling in the training data, anomaly detection can be *supervised*, *unsupervised*, or *semi-supervised*. In *supervised* anomaly detection, training data are labeled accurately and the predictive model is built for different categories, which usually leads to accurate detection of known anomalies. However, obtaining a representative set of accurately labeled data is challenging, especially when data are dynamic and evolving over time. *Unsupervised* anomaly detection handles the most frequent real-world scenarios where labeled data are missing. It naturally applies to dynamic data and is able to detect rare or zero-day anomalies. Unlike in supervised techniques, the chance of overfitting is lower using unsupervised anomaly detection. Finally, *semi-supervised* anomaly detection is a compromise between the supervised and unsupervised techniques, and it requires labeling of only normal categories in the training data, and therefore, building the normal behavior model. Anomaly detection techniques can be divided into the following categories: classification based, clustering based, nearest neighbor (NN) based, statistical, information theoretic, and spectral [CBK09].

Classification-based anomaly detection is a supervised two-phase approach that, by using the training set, learns a classifier that distinguishes between normal and anomalous test samples. It can belong to one of the following categories: neural networks based, support vector machine based, Bayesian networks based, and rule based [CBK09].

Supervised anomaly detection can also be based on NN analysis, which assumes that normal data are located in dense neighborhoods, while anomalies appear far from their closest neighbors. These techniques can be grouped into distance- and density-based approaches. In the basic NN anomaly detection methods, the anomaly score for detecting an anomalous test sample is defined as the distance to its k-th NN in the training set. This basic method has been extended in terms of how the anomaly score is defined and what distance measure is used, as well as to improve the efficiency. In density-based techniques, the density of a sample neighborhood is estimated; that is, samples that lie in a dense neighborhood are considered to be normal, while samples that lie in a neighborhood with low density are declared to be anomalous. Density-based techniques do not perform well when there are regions of different densities in the data. To overcome this issue, a number of techniques have been developed to compute the density of instances relative to the density of their neighbors, including local outlier factor (LOF) [BKNS00] and influenced outlierness (INFLO) [JTHW06].

Clustering-based anomaly detection can operate in the unsupervised and semi-supervised modes. Although anomaly detection pursues a different goal than clustering, several clustering techniques have been employed for detecting anomalies. Moreover, many clustering algorithms already include the notion of outliers. However, algorithms of this type are not developed specifically to catch anomalies in the data, as their primary goal is to find clusters.

Clustering-based anomaly detection techniques can be grouped into three categories. *Cluster association*-based techniques assume that normal data points belong to clusters in the data. Anomalies are then discovered as the points that do not belong to any cluster. The most famous algorithm of this type is density-based spatial clustering of applications with noise (DBSCAN) [EKSX96]. *Centroid*-based techniques assume that normal data points are located near to their closest cluster centroid, while anomalies are far away. The distance from each point to its closest cluster centroid is used as the anomaly score. For example, self-organizing map (SOM) [Koh98], EM, and *k*-means have been applied for detecting anomalies in several applications [SBE$^+$02]. In [PVI], a dependence clustering (DC)-based approach [PL14] was employed for detecting anomalous users during the authentication process. In *cluster density*-based techniques (e.g., Find-CBLOF [HXD03]), normal data points belong to large and dense clusters, while anomalies belong to small or sparse clusters. Therefore, a data point that does not belong to any cluster, which has a size and/or density greater than a certain threshold, is marked as anomalous.

Statistical anomaly detection techniques estimate the probability regions of a stochastic model; hence, normal data points occur in the high-probability regions, while anomalies occur in the low-probability regions. Statistical algorithms fit a statistical model for normal behavior using the given data. Then, based on the results of a statistical inference test, a new point is classified as normal or anomalous depending on how probable this point is according to the model. In [PVII], a statistical model based on Markov chain was built to detect anomalous traffic.

Information-theoretic anomaly detection techniques define anomalies as instances that generate irregularities in the information content of the dataset. The information content of the data is analyzed using information-theoretic measures, such as entropy or Kolomogorov complexity [LV97]. *Spectral* anomaly detection techniques aim to approximate/embed the data in lower dimensional subspaces where anomalies can easily be detected.

2.3 Model selection

2.3.1 The bias/variance trade-off

In predictive models, the accuracy can decrease due to *overfitting* or *underfitting* the data the models are trained on. *Overfitting* occurs when a predictive model is trained to fit the training data too closely. *Underfitting* happens when a chosen predictive model is not complex enough to capture important features, for example, by using a linear model when a quadratic is necessary. To evaluate model performance better, understanding of the source of prediction errors is important. Prediction errors can be divided into errors due to *bias*, which corresponds to underfitting, and *variance*, which corresponds to overfitting. In real life, due to the imperfect models and finite data, there is always a trade-off between minimization of bias and variance, which is commonly referred to as the bias-variance dilemma. The bias and variance are side effects of the model complexity. In practice, the model complexity should be adjusted so that it is not too low or too high. Low complexity results in poor accuracy, and therefore, high error in both training and test data. In contrast, being able to model training data too well and lacking generalization to test data, high complexity models cause a lower training error and higher test error. In other words, the bias/variance trade-off is represented by the ability of a model to fit the data well, and more importantly, to generalize to unseen data points.

From the mathematical point of view, the bias/variance trade-off can be explained as follows. Suppose we are engaged in regression analysis setting and the training data are assumed to be generated by a function $y = f(\mathbf{x}) + \epsilon$, where $f(\mathbf{x})$ is the ideal function we want to model and ϵ is the random error corresponding to noise, which is distributed normally, with $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$. Machine learning algorithms aim to reconstruct an approximation $\hat{f}(\mathbf{x})$ of the ideal function $f(\mathbf{x})$ from the noisy data y. The expected squared prediction error at a point $\mathbf{x}$ is defined as

$$Err(\mathbf{x}) = E[(y - \hat{f}(\mathbf{x}))^2]. \tag{13}$$

The error (13) can be decomposed into bias and variance components, as follows [HTF09]:

$$Err(\mathbf{x}) = (E[\hat{f}(\mathbf{x})] - f(\mathbf{x}))^2 + E[\hat{f}(\mathbf{x}) - E[\hat{f}(\mathbf{x})]^2] + \sigma^2,$$

which is equivalent to

$$Err(\mathbf{x}) = \text{Bias}^2 + \text{Variance} + \text{Irreducible error}.$$

The irreducible error term σ^2, which is the variance of the new test target, comes from the noise ϵ in the training data and is a lower bound for the mean squared error. More details related to deriving $Err(\mathbf{x})$ can be found in [HTF09]. Bias does not depend on the size of the training set, and it is high when the data distribution cannot be modeled well by the selected classifier. Variance depends on the

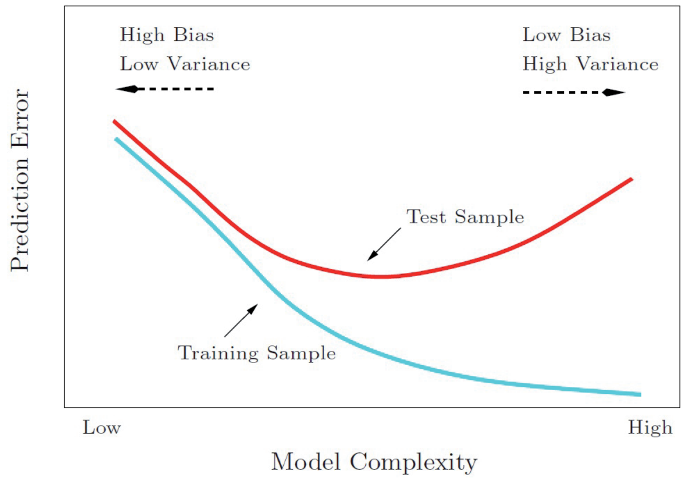


FIGURE 4 Test and training error as a function of model complexity from [HTF09].

training set size. The more data are in the training set, the lower the variance will be. Figure 4 [HTF09] shows how the prediction error depends on the model complexity. As the model complexity increases, the training error and model generalization tend to decrease. In contrast, the low complexity of the model causes underfitting and may result in poor generalization due to a large bias. The variance increases as classifiers become more complicated.

2.3.2 Consensus/ensemble approaches

Many machine learning techniques are stochastic by nature and can benefit from combining the results of multiple models. These models can be trained using different random seeds, different randomly chosen subsets/combinations of features, and various initial conditions/parameters; moreover, they can be provided as outputs of a number of algorithms. The results are then combined according to some consensus criteria.

A combination of the consensus approach, also referred to as the ensemble approach, and popular existing classification or clustering techniques leads to more accurate partitions compared with the results of single learners. Ensemble approaches lead to reduced variance, as the results are less dependent on peculiarities of a single training set, and reduced bias, as a combination of multiple classifiers may learn a more expressive concept class than a single classifier can. Among the most popular ensemble approaches are bagging and boosting.

2.3.2.1 Bagging

Bagging, or bootstrap aggregation [Bre96], reduces variance by averaging the prediction over a collection of bootstrap samples. Bootstrap samples are additional data for training generated from the original dataset using random combinations with repetitions. By training classifiers on a repeatedly perturbed training set, bagging generates multiple predictors. The results of the predictors are combined by voting, for the task of classification, or averaging, for the task of regression. By increasing the size of the training set, one can decrease the variance, leaving the bias unchanged, which leads to improved prediction. Bagging is usually employed with regression or classification trees, but it can be used with other methods as well. There are variations of the classical bagging procedure, which include sampling of features instead of data instances or learning a set of classifiers with different algorithms.

2.3.2.2 Boosting

Boosting [FS99], initially designed for classification problems, can be extended to regression as well. Boosting is a two-step approach that combines the outputs of several weak classifiers to produce a powerful consensus solution. First, one uses subsets of the original data to produce a series of moderately well-performing models, or weak classifiers. Next, the performance of the weak classifiers is boosted by combining them using a specified cost function. In the classical boosting, every new subset contains the elements that were misclassified by the previous models. However, boosting is not limited to being used with weak learners. It can also be used with accurate classifiers, thereby serving as the base learner. Boosting can significantly lower the error rate of a method, as it decreases variance without increasing bias.

2.4 Big data

Modern data communication and social networks have been growing considerably in size, variety, and complexity. Along with the growing complexity of the existing networks, new types of communication networks have emerged. To name a few, Internet of Things (IoT), cloud-based, multi-agent, and wireless-based networks have gained popularity among researchers and companies. The rapid expansion of big data has been accelerated by the dramatic increase of the Internet usage, development of cloud computing technologies, acceptance of social networking and smartphone applications, and so on. There are many applications of big data in different areas, including health, business, and technology [FB13], [LRU14].

Big data is usually associated with large and complex datasets characterized by the following key dimensions [DGdLM13]:

– *Volume*: a vast amount of data is constantly generated, posing challenges for storage and analysis using traditional techniques;
– *Velocity*: data are generated continuously at an exponential rate;
– *Variety*: generated data have different formats, with the majority being unstructured; thus, modern techniques must be adapted to handling heterogeneous data;

- *Veracity*: data may not have sufficient quality or trustworthiness; and
- *Value*: huge amounts of data do not necessarily translate into high-value data, as turning data into value (identifying what is valuable with further transformation, extraction, and analysis) is challenging.

Data are generated at an exponential rate around the world, and this has become a major source of generating profit increases for many organizations; numerous emerging companies work solely around data technology. Therefore, businesses and researchers are focusing on developing new efficient technologies and architectures for processing huge amounts of data and extracting value from them. With the increase of data in terms of volume, velocity, variety, veracity, and value, traditional techniques suffer from limitations related to the efficient storing, processing, and analyzing of the data. To extract meaning from these data, novel and evolving algorithms and analytics techniques have emerged along with innovative and effective ways to use hardware and software platforms. Combining recent achievements in data mining and large-scale computation technologies results in innovation in algorithms for the processing and analysis of big datasets. A common approach is to sacrifice accuracy for efficiency. By using smart approximations, it is possible to increase efficiency dramatically, while the accuracy lost is marginal.

Processing large-scale data is an important problem for many domains. Efficient analysis and timely processing of big datasets requires scalable algorithms and computational frameworks. Traditional solutions are time consuming; hence, new technologies and data processing frameworks supporting the big data phenomenon have recently emerged, to name a few, NoSQL, parallel and distributed paradigms, Apache Hadoop, and Apache Spark.

Apache Hadoop

Hadoop [bib] is an open-source implementation of the MapReduce model [DG08], and it is one of the most widely used frameworks/platforms for the distributed processing of large-scale data. Apache Hadoop implements a scalable fault-tolerant distributed platform for MapReduce programs. The Hadoop Distributed File System (HDFS) provides high-throughput access to application data. Hadoop is reliable and efficient for big data analysis on large clusters. The main concern about Hadoop is maintaining the speed in processing large datasets. Because the data are processed from a disk, Hadoop is inefficient for data mining applications, which often require numerous iterations, as the waiting time between queries and waiting time to run the program take too long. Moreover, despite performing relatively well for offline data, it handles real-time stream data poorly.

Apache Spark

Apache Spark [ZCD+12] is a more recent open-source distributed framework for data analytics that enables, among other things, fast and efficient processing of large streams of data. The key features of Spark are in-memory compu-

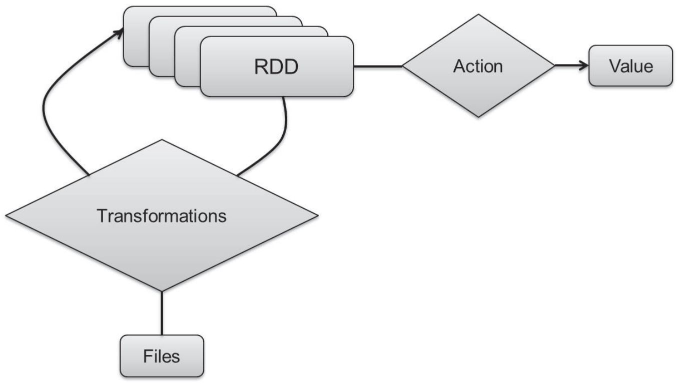


FIGURE 5 Apache Spark RDD workflow.

tations that increase the processing speed of an application and fault tolerance. In addition, Spark supports in-memory caching of datasets, which prevents slow disk reads and reduces network communications, and as a result, performs much faster compared with Hadoop-like systems.

Spark adopts Resilient Distributed Dataset (RDD) [ZCD$^+$12], a distributed memory abstraction that supports two types of operations, as follows: transformations and actions. Transformations define a new RDD based on the existing one, and actions either return a value to the driver program or export data to persistent storage. When a transformation is executed, a new RDD is created, with its records distributed across the main memory. An action operation causes each node to process its local set of records and return the result. The Spark RDD workflow is illustrated in Figure 5.

Spark is used as a tool for creating competent solutions for big data analysis by performing machine learning and data mining tasks. Many clustering and anomaly-detection algorithms have been developed or adapted to Spark due to its efficiency and high performance [MBY$^+$15], [HMX$^+$15],[BTM16],[LC10]. In [PV], a scalable version of the DC algorithm was proposed and implemented in Apache Spark using GraphX application programming interface (API) primitives [XGFS13], [XCD$^+$14].

3 RESEARCH CONTRIBUTIONS

This chapter presents as overview of the results, research contributions and author's contribution to the included articles. The chapter is divided into sections according to the problem areas addressed in this work, which helps to reflect the contributions in a more comprehensive fashion.

3.1 Variable selection

Articles [PI] and [PII] refer to the problem of variable selection in data-driven modeling. Specifically, they consider the papermaking industry, where the changes in paper designs were tested using a pilot paper machine.

Article [PI]: Information-theoretic approach to variable selection in predictive models applied to paper machine data

Article [PI] presents a pilot paper machine case study of variable selection for predicting laboratory measurements of paper quality. The quality of the final paper product depends on numerous quality and process variables in a complicated and nonlinear way. In reality, small changes in process variables can lead to global changes in quality variables. Field testing is usually a long lasting and expensive process. Therefore, it is highly important to establish the links among process and quality variables, that is, determine which process variables influence the quality variables the most. In this paper, principal component analysis (PCA) [Jol02], Shannon mutual information [Sha48], and maximal information coefficient (MIC) [RRF$^+$11] based variable selection techniques were utilized in the preprocessing phase. Then, the MLP model was applied to the selected subsets of process variables to predict measurements for the three quality variables classified as the most important ones according to expert knowledge. Based on the prediction accuracy, the effect of the input variable selection technique on the paper quality prediction was analyzed. The three chosen methods for variable selection were tested and compared using real data collected from the pilot paper machine. Despite being a relatively new concept, MIC produced the best results, while PCA appeared to be the least accurate among the three methods (see Table II, Figure 1 in [PI]). Therefore, the suggested information-theoretic approach to variable selection not only allowed accurate prediction results when working with real data, but it also provided direct insight into the relationships between variables.

The author of this thesis is the main author of this article. She preprocessed the data, implemented the testing procedure, performed the data analysis, and wrote most of the paper.

Article [PII]: Variable group selection based on regression trees: Paper machine case study

Article [PII] presents another pilot paper machine case study, which is a continuation of the work presented in [PI]. In this paper, a regression tree-based methodology for selecting the best groups of predictor variables is proposed. Testing operation scenarios is an expensive process, which makes collecting extra data challenging and often results in situations where features dominate over data instances. The discovery of a sufficiently small subset of the strongest correlations among process and quality variables can provide a better insight into industrial processes, thereby facilitating the modeling procedure. To increase the efficiency of the process of variable selection, data analysis algorithms combined with expert knowledge can be used. This paper focuses on discovering pairs of process variables that demonstrate the strongest correlations with quality variables when considered jointly, yet have small or no correlations between them.

The authors compared two methods of selecting the process variable pairs that are best correlated with quality variables, which were based on regression tree analysis. In both approaches, models were trained and validated on 100 training sets, which were randomly subsampled from the data, using 10-fold cross validation. The trees were pruned according to the validation results. The approaches differ in the way how they identified the most important process variable pairs. The first approach suggested choosing the most important variable pairs based on the frequency of their occurrence in the top two levels of the trees with highest accuracy over 100 models. The second approach stated that the total effect of a variable can be distributed across the tree. First, the importance scores were calculated for each unique variable in each of the 100 trained regression tree models. Then, a pair of variables with the highest individual importance scores was marked as a candidate pair. Finally, the most important variable pairs were selected as the most frequent ones among the candidate pairs. The outcomes of the proposed methodology can be seen in Tables II-V in [PII].

The author of this thesis is the main author of this article. She preprocessed the data, implemented the testing procedure, performed the data analysis, and wrote most of the paper.

3.2 Community detection

Articles [PIII], [PIV], and [PV] refer to the problems of clustering and community detection and developing algorithms to increase the detection accuracy and performance efficiency, especially when used for big data analysis.

Article [PIII]: Revealing Community Structures by Ensemble Clustering using Group Diffusion

In article [PIII], an ensemble clustering approach using group diffusion is proposed to reveal community structures in data. As stated in Section 2.2.2, the community detection task is similar to clustering. The main difference between these approaches is that the community detection algorithms analyze the network structure starting with relationships between objects as input, and therefore, they are applied in complex systems represented as graphs. The proposed algorithm, called ensemble group diffusion (EGD), takes into account the geometric structure of data using group distances and their diffusion across connectivity scales. Depending on the value of the diffusion-depth parameter, the presented approach can accurately identify local clusters and the global structure of the data. Therefore, EGD can determine the underlying geometry of the data, even for those datasets where some of the common clustering methods fail (see Figure 6 in [PIII]). The method is also capable of reasonably regulating cluster sizes by an admitted group dependence level. Moreover, EGD is able to produce more robust clustering results by collectively integrating views from individual diffusion scales. In this way, a reduction of the combined effect of both bias and variance error components is expected. In addition, it handles directed graphs nicely.

The proposed EGD algorithm was tested under different settings using both simulated and real-world datasets against selected state-of-the-art methods of different types that are frequently used for community structure detection, that is, modularity clustering, hierarchical clustering, spectral clustering, density-based clustering [RL14], and an ensemble clustering approach using a knowledge-reuse framework [SG03]. The tests show that due to flexibility in the parameter setting, EGD is suitable for solving structure discovery problems for datasets with different underlying structural and density properties. Thus, EGD can discover local clusters with smaller values of the diffusion depth parameter, while the global structure can be determined with the higher parameter values (e.g., see Figure 4 in [PIII]). The combination of the diffusion depth values allows clusters to be defined more accurately. The performance test summary for all the methods including EGD across nine datasets can be seen in Figure 7 in [PIII]. More detailed performance results can be found in supplementary Tables 4-9 in [PIII].

The author of this thesis developed and implemented the proposed algorithm based on the DC method [PL14]; performed most of the tests; interpreted most of the results; and produced all the figures and tables (except Figure 7). Moreover, she performed literature review; and wrote Sections 1, 2, 4.1 (partially), 4.2, 5 (except 5.4), and 6.

Article [PIV]: Probabilistic group dependence approach for discovering overlapping clusters

To extend the work in [PIII], paper [PIV] proposes a soft dependence clustering (SDC) algorithm that is a generalization of the DC method and supports soft clustering by introducing a soft probability interval. The method adeptly handles different types of data from different domains, such as biology and text data. For text datasets, it proved to be more accurate in detecting multiple groups compared to latent Dirichlet allocation (LDA) [BNJ03], which is a method specifically designed for clustering text data. SDC can be used in graph segmentation applications, as well as for clustering data that have a notion of distance.

The author of the thesis is the main author of this publication. She developed and implemented the proposed algorithm based on DC, performed most of the tests, interpreted the results, produced all the figures and tables, wrote most of the paper, and presented the paper at the 26th International Workshop on Machine Learning for Signal Processing in Vietri sul mare, Salerno, Italy.

Article [PV]: Scalable implementation of dependence clustering in Apache Spark

Article [PV] proposes a scalable version of the DC algorithm that allows better performance in the analysis of big datasets. The method is implemented in Apache Spark using GraphX API primitives. Moreover, a fast approximate diffusion procedure that enables algorithms of the spectral clustering type in the Spark environment is introduced and implemented in Apache Spark. The proposed implementation was tested using real data presented as densely connected graphs. The method proved to scale well, and it was more accurate than spectral clustering.

The author of the thesis is the only author of this publication. She developed and implemented the proposed algorithm, performed the tests, interpreted the results, produced all the figures and tables, wrote the paper, and presented it at the Conference on Evolving and Adaptive Intelligent Systems in Ljubljana, Slovenia.

3.3 Anomaly detection

Anomaly detection has many applications in various domains. This section reflects on the thesis contributions in the field of security, that is, user authentication in [PVI] and network intrusion detection in [PVII].

Article [PVI]: Anomaly detection approach to keystroke dynamics based user authentication

Paper [PVI] proposes two anomaly detection approaches to user authentication using keystroke dynamics based on *k*NN and DC. In this paper, the DC algorithm previously used for the clustering tasks only was adapted to solving anomaly detection problems by introducing an anomaly score measure. Thus, to detect anomalies, the training set was first clustered using DC. Then, the test set was classified according to the median Manhattan distance between a test sample and each cluster mean (anomaly score). In addition, a *k*NN-based approach was applied. As mentioned in Section 2.1.3, *k*NN is a simple and strong benchmark, and it often works well given the right distance measure. In [PVI], a *k*NN-based approach combined with Manhattan distance was employed. Both proposed approaches were tested and compared with multiple state-of-the-art classifiers, and they demonstrated improved results for the CMU keystroke dynamics benchmark dataset reported in [KM09], [ZDJ12].

The author of this thesis is the main author of this article. She preprocessed the data, developed and implemented the proposed approaches and testing procedure, performed the data analysis, interpreted the results, and wrote most of the paper. The author of this thesis also presented the paper at the 22nd IEEE Symposium on Computers and Communications in Heraklion, Creete, Greece.

Article [PVII]: Probabilistic Transition-Based Approach for Detecting Application-Layer DDoS Attacks in Encrypted Software-Defined Networks

Article [PVII] presents a real-time probabilistifc transition-based approach for detecting application-layer distributed denial-of-service (DDoS) attacks in encrypted software-defined networks. Operating with information extracted from packet headers, the proposed solution can be applied without decrypting in secure protocols that encrypt network connections data. In addition, a DDoS detection system prototype has been implemented for evaluating the proposed approach. The proposed attack detection mechanism comprises two phases. First, elementary user behavior patterns are built by applying a clustering algorithm. Next, a probabilistic transition-based approach is applied to the outcomes of the first phase to discover more complex sequential behavior patterns. The enhancements of the proposed approach compared with the previous works [ZKHS16] are as follows: (1) improved performance scores (see Table 1, Figure 2 in [PVII]), (2) a reduced number of parameters, and (3) a significant reduction in the amount of storage needed by the detection algorithm.

The author of this thesis developed and implemented the proposed probabilistic transition-based approach, performed the testing and evaluation procedures, interpreted the results, and wrote most of the paper. She also presented the paper at the 11th International Conference on Network and System Security in Helsinki.

4 CONCLUSION

The main achievements of the work reported in this thesis can be summarized as follows:

– Approaches based on mutual information measure and regression trees were proposed for the variable selection task to improve the prediction models in paper quality control processes. When the dataset is such that features dominate over samples, the proper selection of the most important predictor variables is essential for the model generalization and prevention of overfitting;
– Advanced variants of the DC algorithm were developed in the following ways: (1) to improve the model's stability, DC was extended to an ensemble version, (2) an approach to handling overlapping clusters was developed to address the need for better handling of the overlapping nature of many network communities, (3) an adaptation for solving anomaly detection problems was introduced, and (4) DC was implemented and adapted to the Apache Spark framework, opening the ability to work with big datasets.
– A real-time probabilistic transition-based approach proposed for detecting application-layer DDoS attacks demonstrated significant improvements over the previous works. These advancements included improved performance scores, a reduced number of parameters, and a significant reduction in the amount of storage needed by the detection algorithm, which are critical characteristics when applied in real-time systems; and
– The applications of the developed methods to real datasets showed significant performance gains compared with the reference techniques.

In the next stage of the research, the proposed real-time probabilistic transition-based approach for detecting application-layer DDoS attacks can be implemented and tested in a real-time system in a natural setting. Furthermore, other machine learning approaches, such as deep learning [GBC16], which has gained tremendous popularity, can be employed as an attempt to address problems of, for example, anomaly detection in network security.

YHTEENVETO (FINNISH SUMMARY)

Älykkäitä itseoppivia ratkaisuja reaalimaailman tietopohjaisille sovelluksille

Koneoppiminen on tehokas työkalu suurta tietojenkäsittelyä vaativissa tehtävissä, joissa oman ohjelman kirjoittaminen ongelman ratkaisemiseksi on hyvin kompleksista tai jopa mahdotonta. Tässä mielessä koneoppiminen voidaan katsoa lähestymistavaksi, joka oppii ja tekee itse ohjelmia käsittelemistään tiedoista. Kaikkien koneoppimisalgoritmien yhteinen tavoite on kerätä hyödyllistä tietoa datasta ja hyödyntää sitä ongelmien ratkaisemiseksi. Koneoppimisalgoritmit voidaan jakaa karkeasti kahteen kategoriaan eli valvottuun ja ei-valvottuun koneoppimiseen. Valvottua oppimista käytetään ennakointi- ja luokittelutehtävissä, kun oppimiseen liittyvät esimerkkitunnisteet tunnetaan etukäteen. Tämä vastaa tilannetta, jossa sekä syötteet että tulosteet annetaan ohjelmalle. Ei-valvomaton oppiminen viittaa tilanteeseen, jossa löydetään yleiset rakenteet datasta, esimerkiksi klusteroimalla tai poikkeavuuksien havaitsemisella tilanteissa, joissa tunnisteita ei ole ennakkoon saatavilla.

Tämän väitöskirjan tarkoituksena on kehittää uusia menetelmiä regressioanalyysille, klusteroinnille ja poikkeamien havaitsemiselle. Teollisuus etsii jatkuvasti parempia menetelmiä tuotantoonsa ja kustannustensa minimointiin ja suurten tietomassojen ollessa kyseessä koneoppiminen on yksi hyvä työkalu siihen. Tähän liittyen tässä työssä esitetään useita tapaustutkimuksia, joissa ehdotetaan matemaattisia malleja paperin laadun ennustamiseksi. Tärkeimmät ennustemallien muuttujat on valittu informaatioteorian ja regressiopuiden avulla. Väitöskirjan muut julkaisut kohdistuvat ilman valvontaa tapahtuvaan koneoppimiseen. Pääpaino on kehittyneiden spektristen klusterointitekniikoiden kehittäminen erilaisten yhteisöjen ja poikkeavuuksien havaitsemisessa. Osana tätä työtä ehdotetaan useita parannuksia riippuvuusklusterointialgoritmeihin. Esitettyjä parannuksia ovat muun muassa regularisoinnin lisääminen klustereiden kokoa ajatellen, kokoonpanon laajentaminen mallin vakauden parantamiseksi, päällekkäisten klustereiden käsittely, sekä mallit anomalioiden havaitsemisongelmiin ja suurien tietojoukkojen käsittelyyn.

Työn yksi painopiste on ollut poikkeamienhavainnointialgoritmien kehittäminen tietoverkoissa. Tässä yhteydessä työssä esitetään todennäköisyyteen perustuva siirtymäpohjainen lähestymistapa sovelluskerroksen hajautetun palvelunestohyökkäyksen havaitsemiseksi. Kehitettyjä lähestymistapoja on testattu todellisella verkkoliikenteellä ja ne kykenevät tehokkaisiin ratkaisuihin korkealla tarkkuudella ja hyvillä tuloksilla. Väitöskirjassa esitetyt menetelmät ovat sovellettavissa muuttujan valinnan, kaavion segmentoinnin ja poikkeamien havainnointiin erilaisissa sovelluksissa.

REFERENCES

[ABFX08] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[ABL10] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[Bez81] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[BGK$^+$05] Jeffrey Baumes, Mark K. Goldberg, Mukkai S. Krishnamoorthy, Malik Magdon-Ismail, and Nathan Preston. Finding communities by clustering a graph into overlapping subgraphs. In *IADIS AC*, pages 97–104. IADIS, 2005.

[bib] Apache hadoop. Available at http://hadoop.apache.org/.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[Bre96] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[BTM16] Neha Bharill, Aruna Tiwari, and Aayushi Malviya. Fuzzy based clustering algorithms to handle big data with implementation on apache spark. In *Second IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2016, Oxford, United Kingdom, March 29 - April 1, 2016*, pages 95–104, 2016.

[BZM07] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, July 2009.

[CEB$^+$07] D. Richard Cutler, Thomas C. Edwards, Karen H. Beard, Adele Cutler, Kyle T. Hess, Jacob Gibson, and Joshua J. Lawler. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

[CMP03] C. I. Christodoulou, S. C. Michaelides, and C. S. Pattichis. Multifeature texture analysis for the classification of clouds in satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 41(11):2662–2668, Nov 2003.

[DF10] G. De'ath and K. E. Fabricius. Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81:3178–3192, 2010.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. volume 51, pages 107–113, New York, NY, USA, January 2008. ACM.

[DGdLM13] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey. Addressing big data issues in scientific data infrastructure. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 48–55, May 2013.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[DM04] Luca Donetti and Miguel A Muñoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, 2004.

[DWM02] Jennifer A. Dunne, Richard J. Williams, and Neo D. Martinez. Food-web structure and network theory: The role of connectance and size. *Proceedings of the National Academy of Sciences*, 99(20):12917–12922, 2002.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and*, pages 226–231, 1996.

[FB07] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007.

[FB13] Wei Fan and Albert Bifet. Mining big data: Current status, and forecast to the future. *SIGKDD Explor. Newsl.*, 14(2):1–5, April 2013.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[FJBD06] Alexandru Floares, Angela Jakary, Aaron Bornstein, and Raymond Deicken. Neural networks and classification & regression trees are able to distinguish female with major depression from healhy controls using neuroimaging data. In *IJCNN*, pages 4605–4611, 2006.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010.

[Fre77] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[FS99] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GBS06] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R. Sveinsson. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294 – 300, 2006. Pattern Recognition in Remote Sensing (PRRS 2004).

[GN02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[Hay11] S.O. Haykin. *Neural Networks and Learning Machines*. Pearson Education, 2011.

[HK92] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, Sep 1992.

[HK07] Andreas Hecker and Thomas Kürner. Application of classification and regression trees for paging traffic prediction in lac planning. In *VTC Spring*, pages 874–878. IEEE, 2007.

[HMX$^+$15] Liu Haoxi, Dong Min, Tang Xue, Bi Sheng, Cao Dan, and Qiu Rongcai. The spark-based framework for mobile network data and cluster analysis on mobile users' behaviors. In *2015 IEEE International Conference on Robotics and Biomimetics, ROBIO 2015, Zhuhai, China, December 6-9, 2015*, pages 487–492, 2015.

[HSL$^+$11] Jianbin Huang, Heli Sun, Yaguang Liu, Qinbao Song, and Tim Weninger. Towards online multiresolution community detection in large-scale networks. *PLOS ONE*, 6(8):1–11, 08 2011.

[HSM01] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of Data Mining*. MIT Press, Cambridge, MA, USA, 2001.

[HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Ann. Statist.*, 36(3):1171–1220, 06 2008.

[HTF09] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2009.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recogn. Lett.*, 24(9-10):1641–1650, June 2003.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Proceedings*, pages 448–456. JMLR.org, 2015.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[Jol02] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[JTA$^+$00] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[JTHW06] Wen Jin, Anthony K. H. Tung, Jiawei Han, and Wei Wang. *Ranking Outliers Using Symmetric Neighborhood Relationship*, pages 577–593. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[KAKN06] S. Karimifard, A. Ahmadian, M. Khoshnevisan, and M. S. Nambakhsh. Morphological heart arrhythmia detection using hermitian basis functions and knn classifier. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1367–1370, Aug 2006.

[KM09] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *DSN*, pages 125–134. IEEE Computer Society, 2009.

[Koh98] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.

[LC10] Frank Lin and William W. Cohen. Power iteration clustering. In *ICML*, pages 655–662. Omnipress, 2010.

[Lee91] Yuchun Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3(3):440–449, 1991.

[LF11] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, Dec 2011.

[LKO11] Joo-Ho Lee, In-Yong Kim, and Christine M. O'Keefe. Applicability of regression-tree-based synthetic data methods for business data. In *ICDM Workshops*, pages 651–658. IEEE, 2011.

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2nd edition, 2014.

[Lux07] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[LV97] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (2Nd Ed.)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[MB88] G. J. McLachlan and K. E. Basford. *Mixture models: inference and applications to clustering*. Marcel Dekker Inc, 1988.

[MBY$^+$15] Xiangrui Meng, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. Mllib: Machine learning in apache spark. *CoRR*, abs/1505.06807, 2015.

[MNW02] Ronald E McRoberts, Mark D Nelson, and Daniel G Wendt. Stratified estimation of forest area using satellite imagery, inventory data, and the k-nearest neighbors technique. *Remote Sensing of Environment*, 82(2):457 – 468, 2002.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MS01] Marina Meila and Jianbo Shi. Learning segmentation by random walks. In *In Advances in Neural Information Processing Systems*, pages 873–879. MIT Press, 2001.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(026113), 2004.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[PDFV05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[PL14] Hyunwoo Park and Kichun Lee. Dependence clustering, a method revealing community structure with group dependence. *Knowledge-Based Systems*, 60:58–72, 2014.

[Pot97] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. Technical report, Norfolk, VA, USA, 1997.

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[RL14] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, June 2014.

[RRF$^+$11] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[SBE$^+$02] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, 2002.

[SC11] John P. Scott and Peter J. Carrington. *The SAGE Handbook of Social Network Analysis*. Sage Publications Ltd., 2011.

[SG03] Alexander Strehl and Joydeep Ghosh. Cluster ensembles-a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, March 2003.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[SHK^{+}14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

[SPB10] Matthew S. Spencer, Samantha C. Bates Prins, and Margaret S. Beckom. Heterogeneous distance measures and nearest-neighbor classification in an ecological setting. *Missouri J. Math. Sci.*, 22(2):108–123, 05 2010.

[SSK13] Indu Saini, Dilbag Singh, and Arun Khosla. Qrs detection using k-nearest neighbor algorithm (knn) and evaluation on standard ecg databases. *Journal of Advanced Research*, 4(4):331 – 344, 2013.

[STB00] H.E.S. Said, T.N. Tan, and K.D. Baker. Personal identification based on handwriting. *Pattern Recognition*, 33(1):149 – 160, 2000.

[SWA08] Alexander Statnikov, Lily Wang, and Constantin F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9(1):319, Jul 2008.

[WM97] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *J. Artif. Int. Res.*, 6(1):1–34, January 1997.

[XCD^{+}14] Reynold S. Xin, Daniel Crankshaw, Ankur Dave, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: Unifying data-parallel and graph-parallel analytics. *CoRR*, abs/1402.2394, 2014.

[XGFS13] Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, GRADES '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM.

[YS09] Xin Yan and Xiao Gang Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009.

[YW99] Y. Yohannes and P. Webb. *Classification and Regression Trees, CART: A User Manual for Identifying Indicators of Vulnerability to Famine and Chronic Food Insecurity*. Microcomputers in policy research. International Food Policy Research Institute, 1999.

[ZBA12] C. Zanchettin, B. L. D. Bezerra, and W. W. Azevedo. A knn-svm hybrid model for cursive handwriting recognition. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, June 2012.

[ZCD+12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[ZDJ12] Yu Zhong, Yunbin Deng, and Anil K. Jain. Keystroke dynamics for user authentication. In *CVPR Workshops*, pages 117–123. IEEE Computer Society, 2012.

[ZKHS16] Mikhail Zolotukhin, Tero Kokkonen, Timo Hämäläinen, and Jarmo Siltanen. On application-layer ddos attack detection in high-speed encrypted networks. *Intl. Journal of Digital Content Technology and its Applications*, 10(5):14–33, 2016.

ORIGINAL PAPERS

PI

INFORMATION-THEORETIC APPROACH TO VARIABLE SELECTION IN PREDICTIVE MODELS APPLIED TO PAPER MACHINE DATA

by

Elena Ivannikova, Timo Hämäläinen, and Kari Luostarinen 2013

Proc. of the IEEE Symposium on Computers and Communications, pp. 946-950

Information-Theoretic Approach to Variable Selection in Predictive Models Applied to Paper Machine Data

Elena Ivannikova*, Timo Hämäläinen*, Kari Luostarinen†
*Department of Mathematical Information Technology,
University of Jyväskylä, Finland
{elena.v.ivannikova, timo.t.hamalainen}@jyu.fi
†Metso Paper, Jyväskylä, Finland
kari.luostarinen@metso.com

***Abstract*—This paper presents an information-theoretic approach to variable selection for prediction of laboratory measurements of paper quality. Along with a well-known Principal Component Analysis we considered techniques for variable selection based on the classical Shannon Mutual Information and a novel Maximal Information Coefficient. A multilayer perceptron neural model was used to predict quality measurements and compare feature selection techniques. The suggested approach was tested on real industrial data obtained form a pilot paper machine. The presented results show that information-theoretic techniques perform better compared to Principal Component Analysis, providing higher accuracy results.**

I. Introduction

The modern paper making industry is characterized by complex processes. Modeling such processes is often difficult using physical models and therefore requires modeling tools such as neural networks and probabilistic predictive models [1], [2]. Changes in designs of processes are usually costly, and real time (online) measurements can help to improve these processes with less expenses.

There are many challenges in paper making industry, among which cost reduction plays one of the central roles. One way to reduce costs is by lowering the quality of raw materials. However, the final product quality must remain tolerable. This can be ensured through testing various designs in a pilot paper machine. A pilot paper machine allows testing different components, various settings of process control parameters, and combinations of raw materials. Such changes affect the quality of the resulting paper product. The measurements collected during testing new machine designs reflect the effect of these changes on the product quality. A more detailed description of the pilot paper machine experiments performed for obtaining the data sets for current research can be found in [3].

The quality of final paper product depends on many quality and process variables. These dependencies are often complicated and nonlinear. In [3], the authors introduced a multilayer perceptron model for predicting laboratory measurements of paper quality. They used Principal Component Analysis (PCA) [4] for reducing input dimensionality. Based on this study, we examine correlations among process and quality variables and the impact of input variables selection on the accuracy of paper quality prediction. The main focus of this paper is to compare feature selection methods and examine how different techniques in the preprocessing phase affect the final result.

This paper is organized as follows. Section II describes the data and how it was preprocessed, Section III is devoted to dimensionality reduction of the input variables. Three methods are described in the corresponding subsections. Section IV is related to a Neural Network model used to compare dimensionality reduction techniques. The results are presented in Section V, and Section VI highlights the conclusions and outlook to future work.

II. Data description and preprocessing

For our task, we used two data sets obtained from the pilot machine. The data sets contain measurements from pilot machine runs encompassing ten days over the period of seven months. Altogether 229 trial points and 10 different mechanical settings are presented in the data. More detailed information about the data can be found in [3].

Online measurements collected from sensors during experiments are represented as trial points. At each trial point, quality measurements and time-averaged process state measurements had been stored. In the current research we consider 186 state variables. As we are interested in the end product quality, we omitted the measurements related to process behavior. Moreover, some measurements which were carried out straight after a change in the mechanical settings are usually biased. Thus, we deleted those trial points which produced unreliable source measurements. In addition, some trials did not contain a full set of measurements, as an interesting behavior had been expected only for several specific paper qualities. Thus, different collections of trial points are available for all the paper qualities and due to this we have to examine each quality variable separately. We also deleted those process variables which were constant or had a variance equaling zero. We generated a separate data set for every quality variable. Table I lists the quality variables examined in this research and the number of trial points considered for every variable. These quality variables can be classified as the most important ones.

TABLE I
QUALITY VARIABLES AND NUMBER OF AVAILABLE TRIAL POINTS.

Quality variable	Unit	Number
Formation Index	(index)	179
Beta-formation	g/m^2	176
Air permeance	ml/min	180

First, the data sets obtained were normalized by z-score normalization, where each value of the obtained numeric matrix Z is calculated from the values of the current matrix X as follows

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}. \quad (1)$$

Here X and Z are $N \times M$ matrixes, N is the number of trial points in a data set, M is the dimensionality of the input space corresponding to a specific quality variable, μ_j is the mean value of the j-th column, and σ_j is the j-th variable standard deviation. At the same time, the features corresponding to quality variables were normalized via $Min - Max$ normalization according to the formula

$$z^*_{ij} = \frac{x_{ij} - \min X(:,j)}{\max X(:,j) - \min X(:,j)}, \quad (2)$$

where Z^* is $N \times M$ matrix and $X(:,j)$ is the j-th column of the matrix X. These normalized data sets have been used further in the analysis.

III. CORRELATION ANALYSIS

Before training a neural model, we start from dimensionality reduction. This step is relevant, as there are likely to be many input variables which correlate among each other, or they are irrelevant or redundant in the context of others. We compare three methods for dimensionality reduction, based on PCA [4], Shannon Mutual Information [5], and Maximal Information Coefficient (MIC) [6]. Sections A, B, and C describe how we applied these techniques.

A. PCA

PCA is a well-known technique. In a way similar to that described in [3], we use a two-step PCA algorithm. Firstly, we project process measurement data on the major principal components that explain 90% of variance. Secondly, we introduce new vectors that contain the squares of the projected coordinates and repeat the procedure from the previous step. The latter step helps to handle nonlinearities in the process measurements data. As a result, 12 coordinates are obtained, which we use as input to the neural network.

B. Mutual Information

The mutual information concept plays an important role in many areas, including data analysis. There are a number of works related to variable selection based on mutual information, e.g. [7]–[9]. Mutual information is based on the concept of entropy of a random variable. The mutual information between two discrete variables X and Y can be interpreted as the amount of information shared by X and Y and reads as follows

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_{x\in\Omega_1}\sum_{y\in\Omega_2} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}, \end{aligned} \quad (3)$$

where entropy $H(X)$ and conditional entropy $H(X|Y)$ are expressed by

$$\begin{aligned} H(X) &= -\sum_{x\in\Omega_1} p(x)\log p(x), \\ H(X|Y) &= -\sum_{y\in\Omega_2} p(y)\sum_{x\in\Omega_1} p(x|y)\log p(x|y). \end{aligned} \quad (4)$$

Here x denotes a possible value of X from the alphabet Ω_1, y denotes a possible value of Y from the alphabet Ω_2 and p represents distribution of a variable. The alphabets Ω_1, Ω_2 have been created by assigning variable values to discrete buckets that quantize the variable domain. Entropy is a measure of uncertainty in the distribution of a random variable. The entropy is low if the outcome of the distribution tends to a particular event, maximal if the outcome is highly uncertain.

For calculation of the mutual information among quality and process variables, we use a framework described in [10]. The calculation of mutual information is based on the Shannon's theory [5]. We calculate the mutual information scores for every pair of quality and process variables, in turn. The higher the mutual information score between a quality and a process variable is, the more valuable this process variable is for predicting the quality variable. From the list obtained, we select 10 variables with the maximal scores. The procedure is repeated separately for every quality variable. Thus, for every quality variable we obtain a list of ten variables with the highest mutual information scores, which we further use as input to the neural network model.

C. MIC

MIC is a measure of dependence and was designed for identifying relationships between pairs of variables in large data sets. This measure is calculated from a matrix of scores generated from a given set of two-variable data. For MIC computation, the largest possible mutual information estimate is calculated for every possible grid resolution encapsulating a relationship between these two variables. Then the MIC is defined as a maximum value among all normalized mutual information estimates achieved by each grid resolution. The process captures different types of interesting relationships, both functional and non-functional. MIC is a part of a family of Maximal Information-based Nonparametric Exploration (MINE) statistics introduced in [6] and can be used to identify and characterize associations between variables. MIC is based on mutual information in a way that mutual information is used to measure performance of each grid. However, it is not an estimate of mutual information.

We apply MINE for identifying associations among variables in our data set. MIC scores are then calculated for every pair of quality and process variables, in turn. We repeat the procedure separately for each quality variable. A MIC score tends to 1 if a never-constant noiseless functional relationship exists between two variables, and the score tends to 0 if the variables are statistically independent. Based on this principle, we then select top 10 process variables, which, in pairs with quality variables, have received the highest MIC scores. We provide the resulting top ten list of process variables as an input for the neural network and predict the quality based on it.

IV. Neural Model

An artificial neural network consists of an interconnected group of artificial neurons which can be divided into several layers [11]. One of the best-known neural models is the multilayer perceptron (MLP). An MLP consists of multiple interconnected layers of nodes comprising a directed graph topology. Each node, except for the input nodes, has an activation function. Neurons of the previous layer provide an input for the neurons of the next layer. Each neuron computes a weighted sum of its inputs and applies the activation function. The weights assigned to connections between neurons serve as free parameters of the neural model. For training the MLP network, we utilize the back-propagation technique with regularization.

Artificial neural networks are powerful tools that have been successfully used to model and solve different problems in paper industry. Neural networks have been used in a number of studies for predicting and estimating paper quality. In [12], the authors propose an MLP network model with one hidden layer to simulate a nonlinear plant process. They use an inverse computation of the network model to find the control settings that guarantee producing the desired quality. In [13], the authors consider an MLP model based on the idea of choosing different preprocessing and training algorithms. They use PCA and stepwise regression statistics for dimensionality reduction in the preprocessing phase and show that a model based on PCA preprocessed data has higher performance. In [14], the authors apply neural network techniques to the prediction of paper "curl". In [3], an MLP neural network model is used for predicting individual quality measurements. The authors train a separate model for each quality variable every time a new data point is added to the history database. In our research, we use a similar approach, but instead of iterative procedure we train the model using the training set and evaluate it using the test set. This research is a follow-up study of [3] with an emphasis on identifying dependencies among the process variables.

Due to the fact that numbers and collections of available trial points are different for every quality variable considered in this research, we train a separate network model for each quality variable. For the MLP modeling, we divided our data set into two parts, 70% for the training and validation set, and 30% for the test set. We split the data in a way that mechanical settings are balanced in the training and test sets. We used jackknife cross validation [15] for validating parameters of the neural network. In jackknife, each instance is consecutively taken out of the training set and predicted from the model built on the remaining instances of the set. Further, we choose the best model in terms of accuracy and perform final testing on the test set.

In this study, an MLP model is used for prediction of laboratory measurements and comparison among the variable-selection techniques. For each quality variable, we apply a multilayer perceptron with three layers of neurons: input, output and hidden. A logistic activation function [11] is used in the hidden layer, and a linear activation function is used in the output layer. The training input vectors that are fed into the neural model are the preprocessed state vectors. The size of the input layer is equal to the dimensionality of the input M. The training targets are the quality laboratory measurements. Thus, the training of the neural network boils down to minimization of the following cost function

$$J(\Theta) = \frac{1}{2N}\left(\sum_{i=1}^{N} |e_i|^2 + \lambda \sum_{j=1}^{L} \Theta_j^2\right). \tag{5}$$

In the formula above, Θ represents the parameters of the model, i.e. connection weights. L equals to the number of connections in the neural network, excluding biases. e_i is prediction error, which is expressed as

$$e_i = \widehat{y_i} - y_i, \tag{6}$$

where $\widehat{y_i}$ represents predicted values, and y_i are target values.

The neural model has two meta-parameters: number of neurons in the hidden level and regularization coefficient λ. The training was performed with 3, 10, and 25 neurons in the hidden layer and with $\lambda \in [0.01, 0.1, 0.2, 0.4]$. After the jackknife cross validation, the parameters were selected in order to guarantee the highest possible prediction accuracy.

V. Results and Conclusions

In this paper, we compare three techniques for variable selection in predictive models. The first technique is the well-known PCA approach, the other two techniques, Mutual Information and MIC, are methods based on the information theory. Table II lists the accuracy values obtained for the prediction of quality measurements using examined variable selection techniques. The accuracy was calculated as

$$P = 1 - \frac{mean(|e|^2)}{var(y)}, \tag{7}$$

where e and y are as denoted in Section IV.

From Table II, one can notice that the predictions for the measurements of the quality variables Air Permeance and Formation Index have been more accurate than the predictions for Beta-formation. This result is similar to the one derived in [3], where this quality variable received the lowest accuracy among a subset of six.

TABLE II
NEURAL MODEL ACCURACY.

Variable selection technique	Quality variable		
	Formation Index	Beta-formation	Air permeance
PCA	64.62	56.33	76.44
MIC	79.01	59.65	88.46
Mutual Information	76.78	59.75	80.04

In addition to the neural model accuracy defined by (7), we used the relative error estimates as a measure of prediction performance. Fig. 1 shows the averaged absolute values of the relative error (in percentages) made by predictions on the test set. These values were calculated as

$$E_i = 100 \times \left| \frac{e_i}{y_i} \right|, \tag{8}$$

where e_i is prediction error defined by (6), and y_i represents target values. From Fig. 1, one can see that based on the relative error measure, MIC appeared to be most accurate, demonstrating the lowest E_i values among the three dimensionality reduction methods for each quality variable.

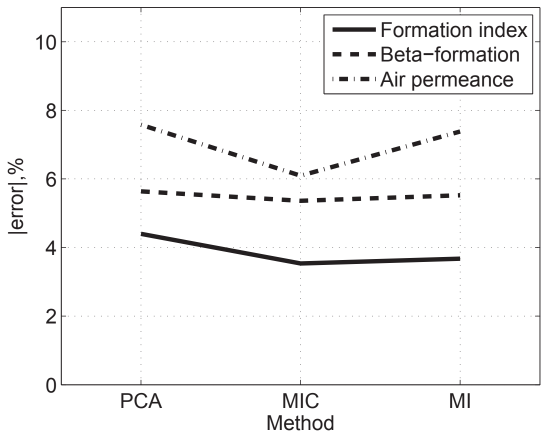


Fig. 1. Average relative error of prediction

The prediction results have been obtained via an MLP neural model applied to nine different data sets. For every quality variable we considered three data sets obtained via three dimensionality reduction techniques. While PCA and Shannon Mutual Information have proven their consistency by being successfully used in many areas, MIC is a relatively new concept. We compared these methods on real data collected from the pilot paper machine using two measures of prediction performance.

Based on the accuracy measure, in two of the three cases MIC performed better, providing higher accuracy results. Whereas according to the relative error measure, MIC demonstrated better results for each quality variable. Overall, both information-theoretic techniques performed well, while PCA was less accurate. Based on the results presented, we can conclude that the information-theoretic approach to variable selection works well with the real data considered in this research. Moreover, in contrast to PCA, the suggested information-theoretic approach gives direct insight into the relationships between variables. This allows us to confirm that the presented approach can be used for solving real problems such as prediction of paper quality and similar tasks. Compared to the traditional PCA technique, this approach improves the predictive quality of the developed models.

VI. DISCUSSION

In this paper, we presented an information-theoretic approach to variables selection for predicting paper quality, based on the measurements from a pilot paper machine. After preprocessing the data, we selected the variables which got the highest mutual information and MIC scores. However, this might not be the optimal way, as there could be correlations among the selected variables and some important variables might be missing. In future studies, we plan to improve the procedure of selecting variables. This could be done by implementing an algorithm which considers correlation among more than two variables. In addition, we plan to compare the performance of the discussed methods to other feature selection techniques based on information theory, such as minimal-redundancy-maximal-relevance criterion [7].

ACKNOWLEDGMENT

The authors would like to thank Dr. Andriy Ivannikov for useful discussions and comments on the paper.

REFERENCES

[1] K. Leiviskä, Ed., *Papermaking science and technology*, 2nd ed., ser. Process and Maintainance Management. Paperi ja Puu Oy, 2009, vol. 14.

[2] G. Bortolin, P. O. Gutman, and B. Nilsson, “On modeling of curl in multi-ply paperboard,” *Journal of Process Control*, vol. 16, pp. 419–429, 2006.

[3] P. Nieminen, T. Krkkinen, K. Luostarinen, and J. Muhonen, “Neural prediction of product quality based on pilot paper machine process measurements,” in *Adaptive and Natural Computing Algorithms*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6593, pp. 240–249.

[4] I. Jollife, *Principal Component Analysis*, 2nd ed. New York: Springer, 2002.

[5] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[6] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, “Detecting novel associations in large data sets,” *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[7] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.

[8] K. Hild, D. Erdogmus, K. Torkkola, and J. Principe, “Feature extraction using information-theoretic learning,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 9, pp. 1385–1392, 2006.

[9] J. B. Ghasemi and E. Zolfonoun, “A new variable selection method based on mutual information maximization by replacing collinear variables for nonlinear quantitative structure-property relationship models,” *Bulletin of the Korean Chemical Society*, vol. 33, pp. 1527–1535, 2012.

[10] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, “Conditional likelihood maximisation: A unifying framework for information theoretic feature selection,” *Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1999.

[12] J. Lampinen and O. Taipale, “Optimization and simulation of quality properties in paper machine with neural networks,” in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 6, 1994, pp. 3812–3815 vol.6.

[13] F. Wang, S. Sanguansintukul, and C. Lursinsap, “Curl forecasting for paper quality in papermaking industry,” in *System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference - 7th International Conference on*, 2008, pp. 1079–1084.

[14] P. Edwards, A. Murray, G. Papadopoulos, A. Wallace, J. Barnard, and G. Smith, “The application of neural networks to the papermaking industry,” *Neural Networks, IEEE Transactions on*, vol. 10, no. 6, pp. 1456–1464, 1999.

[15] H. Abdi and L. J. Williams, “Jackknife,” in *Encyclopedia of Research Design*, N. J. Salkind, Ed. SAGE Publications, Inc., 2010, pp. 656–661.

PII

VARIABLE GROUP SELECTION BASED ON REGRESSION TREES: PAPER MACHINE CASE STUDY

by

Elena Ivannikova, Timo Hämäläinen, and Kari Luostarinen 2014

Proc. of the Conference on Evolving and Adaptive Intelligent Systems, pp. 1-5

Variable Group Selection Based on Regression Trees: Paper Machine Case Study

Elena Ivannikova[*], Timo Hämäläinen[*], Kari Luostarinen[†]
[*]Department of Mathematical Information Technology,
University of Jyväskylä, Finland
{elena.v.ivannikova, timo.t.hamalainen}@jyu.fi
[†]Metso Paper, Jyväskylä, Finland
kari.luostarinen@metso.com

***Abstract*—This paper presents a methodology for selecting best groups of predictor variables based on regression trees. Test results of the developed methodology applied to industrial pilot paper machine data are presented. Specifically, the results list process variable groups, which are more valuable in predicting paper quality variables. The benefit of paper quality prediction based on process variables is the timely reaction to changes happening during production process and, thus, the reduced operational costs. The proposed regression trees based group variable ranking methodology shows stable results on both data sets used in this study.**

***Index Terms*—Prediction, Paper quality, Pilot paper machine, Regression trees.**

I. Introduction

Variable selection is important in data-driven modeling. Thus, expert knowledge can be sufficient for modeling of small systems, as the processes and possible effects are well-known, and different scenarios can be tested and compared interactively. However, as the number of process variables increases, modeling becomes a more complex process. In this scenario, discovery of a sufficiently small subset of strongest correlations among process variables can provide a better insight to process facilitating the modeling procedure.

Variable selection is an important problem especially in those cases when a number of features significantly dominate amount of samples. As an example, let us mention industrial multi-sensor systems with a high number of sensors. In case of pilot paper machine, testing operation scenarios constitutes an expensive process, which makes collecting extra data objectionable and leads to a problem when number of features prevail over instances. Variable selection can also signify a process of selecting a subset of an original set of variables, which sufficiently approximates the whole data set.

A substantial amount of research papers on variable selection and data modeling became recently available. [1] introduces three approaches for variable selection in large-scale industrial systems, among which there are knowledge-based, data-based and model-based methods. Knowledge-based method asserts that using process knowledge can be used in decreasing the number of variables for process modeling. In data-based approach the authors refer to the well-known Principal Component Analysis (PCA) as a dimensionality reduction method. Finally, model-based variable selection is based on the procedure described in [2], where simple dynamic model candidates are constructed systematically for different training data segments. These models are evaluated and the best model is tested on the test set. After modeling, the importance of variables is analyzed based on the occurrence of variables in case models. In [3] the authors analyze the influence of variable selection in the field of traffic signs classification using the Best First [4] attribute selection approach. Evolutionary algorithms reveal another useful tool, which is used in numerous problem areas for solving various tasks including variable selection for spectroscopy, medical chemistry, and others [5], [6].

Specifically, this paper deals with studying variable selection approaches applied to paper making industry. There are many challenges in modern paper making industry, among which cost reduction plays one of the central roles. One way to reduce costs is by lowering the quality of raw materials. However, the final product quality must remain tolerable. Changes in designs of processes are usually costly, and real time measurements can help to improve these processes with less expenses. Various designs are tested in a pilot paper machine, which allows examining different components, various settings of process control parameters, and combinations of raw materials. The measurements collected during testing new machine designs reflect how the changes in parameters and settings affect the product quality. More information about the pilot paper machine experiments performed for collecting the data sets used in this research can be found in [7].

The quality of final paper product for each paper grade and machine design depends on many quality and process variables in a complicated and nonlinear way. Therefore, it is important to know about relationships between process and quality variables. Knowing these allows more efficient selection of the input variables for predicting paper quality. Process knowledge can be used for decreasing the number of variables; however, finding patterns in high-dimensional data is difficult. Therefore, sophisticated data analysis methods can be used for solving this task more efficiently, particularly when combined with expert knowledge. For example, the authors in [1] describe an automated procedure for detecting interactions between the variables in large data sets. The idea of the procedure is based on analysis of the structure properties of the best candidate model selected among all dynamic models candidates. Each model candidate consists of full input combinations for data partitions and is build systematically depending on sliding window size. Another work [7] describes a multilayer perceptron model (MLP) for predicting laboratory measurements of paper quality with the use of PCA [8] for reducing input dimensionality. Following this study, [9] examines correlations among process and quality variables and the impact of input variables selection on the accuracy of paper quality prediction. The authors compare three methods of variable selection based on PCA and mutual information [10]. The authors demonstrate that for the given data sets methods based on mutual information perform better than PCA demonstrating higher accuracy values of the MLP model applied to the selected subsets of variables. However, they point out that the procedure of selecting variables can be improved through considering correlations among more than two variables. In this paper we investigate pairs of process variables, which demonstrate strongest correlations with quality variables, when considered jointly, yet having small or no correlations between themselves.

This paper is organized as follows. Section II describes the data and how it was preprocessed, Section III is devoted to regression trees, which are served as a basis for two methods described in Section IV. The results and conclusions are presented in Sections V, VI.

II. Data description and preprocessing

For our task, we used two data sets obtained from the pilot machine. The data sets contain measurements of the sensors from pilot machine runs encompassing ten days over the period of seven months. The data is expressed as trial points corresponding to changes in process parameters. After a change in process parameters, some time is needed until the process is stabilized. Every trial point usually corresponds to a change in one process parameter, while the rest of the process parameters remain constant. In addition to process parameters, there are mechanical settings, which are the mechanical alterations and which have a major influence on producing the paper. Altogether 229 trial points and 10 different mechanical settings are presented in the data sets. More detailed information about the data can be found in [7].

At each trial point, quality measurements and time-averaged process state measurements had been stored. However, some of the trial points presented in the data were related to process behavior instead of quality of the end product. During preprocessing trial points which were missing laboratory measurements have been omitted since we were interested in the end product quality. Thus, we reduced our data sets to 182 trial points with laboratory results. In addition, some trial points did not contain a full set of measurements, as an interesting behavior had been expected only for several specific paper qualities. Hence, different collections of trial points are available for all the paper qualities and due to this we have to examine each quality variable separately. We also deleted those process variables which were constant or had a variance equaling zero and generated a separate data set for every quality variable.

Further in the research we consider two data sets: (1) full data set and (2) reduced data set. The reduced data set contains only good trial points, which according to the trial leader responsible for the data sets could facilitate the prediction method to work. The good data set means that all unfair or impossible trial points were removed, along with those trial points, which had been gathered in the beginning of data collection or after a mechanical settings change.

Table I lists the quality variables and corresponding units of measurement, classified by expert knowledge as the most important ones and examined in this research. Table I also lists the numbers of trial points containing measurements of each quality variable in the full and reduced data sets.

TABLE I
Quality variables and number of available trial points.

Quality variable	Unit	Number (Full)	Number (Reduced)
Formation Index	(index)	182	148
Beta-formation	g/m^2	178	144
Air permeance	ml/min	182	148

III. Regression trees

Regression analysis builds a predictive model from a training set. Such analysis can have two main purposes: (1) predict a response variable as accurately as possible; (2) understand the structural relationships (patterns) between the response and predictor variables. Classification and regression trees (CART) are an example of regression models used for predicting continuous variables (regression) or categorical variables (classification) [11]. Since in this paper we deal with predicting continuous variables, we consider only regression trees. In regression trees a numerical response is predicted using numerical and categorical predictor variables. A regression tree partitions the feature space X into homogeneous disjoint groups A_k and predicts the most likely fitted value of the dependent variable $E(Y|A_k \in X)$ within each group. Each group is characterized by a typical value of the response variable, a number of samples in the group, and the values of the predictor variables defining it.

The binary tree, considered in this paper, is grown by repeated binary splitting of the data starting from a single node at the top, representing all the data. Each split is defined by a simple binary rule, which is based on a single predictor variable and produces two nodes. The leaves of the tree (unsplit nodes) express groups of data formed by the tree. The splitting continues until a sufficiently rich tree is built. The obtained tree is then usually pruned to offer a required level of generalization.

For the selection of a tree size cross-validation is often used. The best tree is the one having the smallest mean squared error on the validation data set [11].

Regression tree performance is usually given in terms of mean squared error,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y - y^{*})^{2}. \quad (1)$$

Here N, y, y^{*} are number of samples, measured value and predicted value correspondingly.

In regression trees, the input variables can be of different types. The algorithm equally handles numeric, categorical, binary or ordinal variable types. Trees are able to handle missing values and are invariant to changing the relative scales of the predictor variables [11].

Regression trees have been well-studied in the literature with applications in areas including health systems, business, ecology, network planning and many others [12]–[15].

IV. Methods for variable pairs selection

We compare two methods for selection of process variable pairs best correlated with quality variables, based on regression trees analysis. The regression trees model assumes that regression analysis is performed separately for each of the three quality variables and, thus, we repeat the algorithm consequently for each of three quality variables considered in this research. The following subsections describe proposed methods.

A. Reduced regression trees based ranking

The idea of this method can be described as follows. Firstly, we preprocess each of the data sets in a way that 80% of randomly selected samples go to the training and validation set, and 20% remain for the test set. We split the data in a way that mechanical settings are balanced in the training and test sets, as it is known that for every mechanical setting the paper machine is considerably different. Secondly, on the training set we perform the following procedure: (1) training a model; (2) validating the model via 10-fold cross validation, and, finally, (3) pruning the tree based on the validation results.

As the data sets are quite small, we had to handle lack of data by performing final testing on the union of training and test sets. This step allowed us to get a more transparent picture about how the algorithm handles the given data. However, this approach would possibly introduce some bias. Final testing consists in calculating accuracy on the overall data set.

We repeat the procedure starting from splitting the data into training and test sets and ending with calculating accuracy on the overall data set 100 times.

The final step of the algorithm consists of selecting process variables pairs. Firstly, we choose the best models based on the accuracy calculated on the union of the training and test sets.

A model passes best models criterion if its accuracy is above a certain α threshold $A_i > \alpha_i$. Here A_i is the accuracy value of a model at the i-th iteration. Then, the most important variables are chosen according to the frequency of their occurrence in the top two levels of the selected trees with highest accuracy.

TABLE II
Variable pairs on full data set obtained via reduced regression trees based ranking.

Quality variable	Process variable 1	Process variable 2
Formation Index	1st press shoe nip tilt	High-vacuum suction box
Formation Index	1st press bottom roll saveall	Pick-up felt uhle boxes
Formation Index	1st press bottom roll saveall	1st press top roll saveall
Beta-formation	1st press bottom roll saveall	1st press top roll saveall
Air permeance	Suction box 1 chamber 3	3rd press nip water removal
Air permeance	VacuMaster - vacuum	3rd press nip water removal

TABLE III
Variable pairs on reduced data set obtained via Reduced regression trees based ranking.

Quality variable	Process variable 1	Process variable 2
Formation Index	1st press shoe nip tilt	Total slice flow
Formation Index	Wire stretcher roll - power	Starch total dosage
Beta-formation	3rd press bottom felt uhle boxes	Bentonite split Middle
Air permeance	VacuMaster - vacuum	3rd press nip water removal
Air permeance	VacuMaster - vacuum	Total slice flow

Tables II, III list the results of this method. Table II lists the variables obtained after applying the method on the full data set, and Table III represents results acquired on the reduced data set.

B. Regression trees based ranking

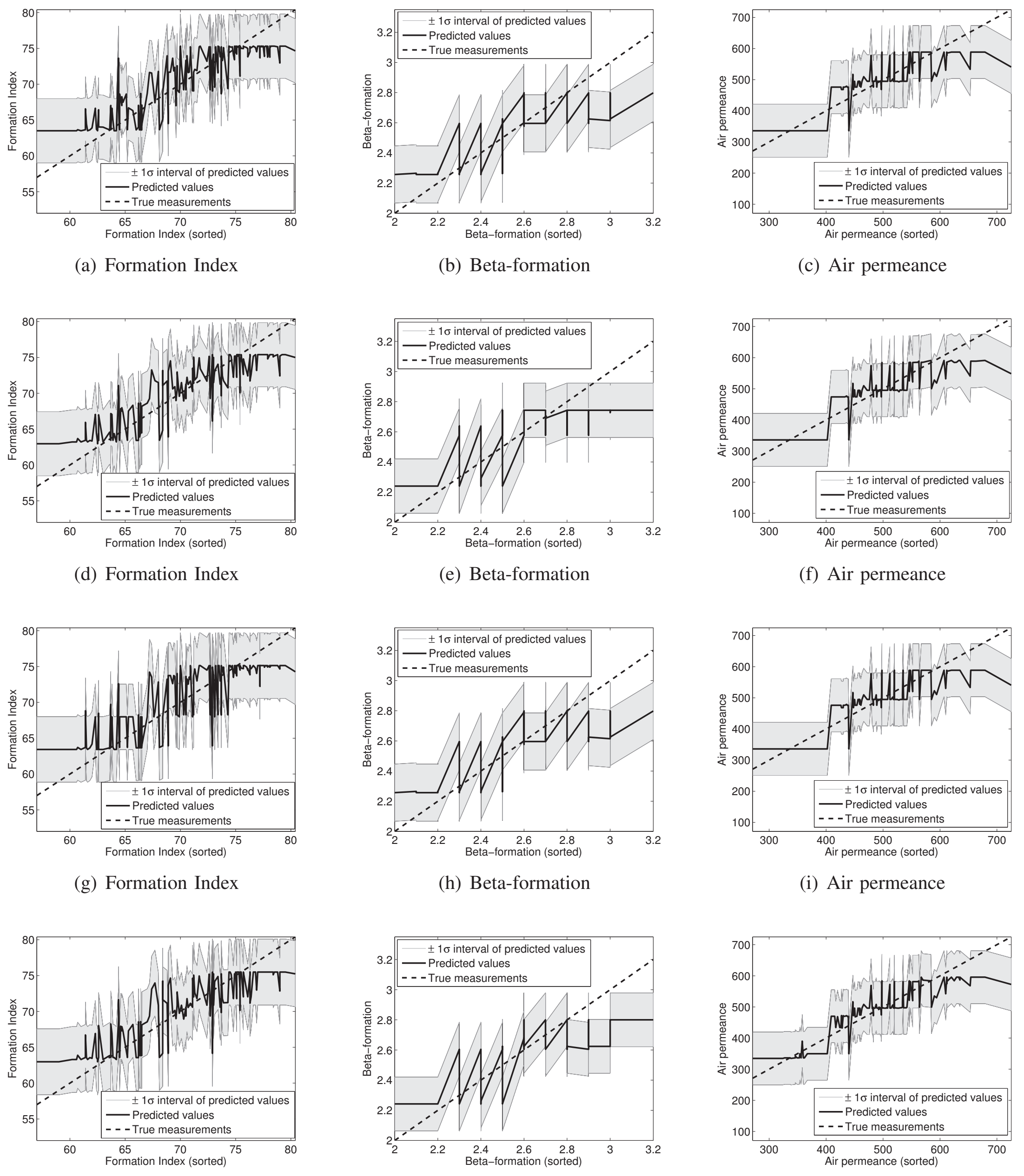
This method is an extension of the reduced regression trees based ranking algorithm described in the previous subsection. In the same way we randomly divide the data into the training

TABLE IV
Variable pairs on full data set obtained via regression trees based ranking.

Quality variable	Process variable 1	Process variable 2
Formation Index	1st press shoe nip tilt	High-vacuum suction box
Beta-formation	1st press bottom roll saveall	1st press top roll saveall
Air permeance	Suction box 1 chamber 3	3rd press nip water removal
Air permeance	VacuMaster - vacuum	3rd press nip water removal

TABLE V
Variable pairs on reduced data set obtained via regression trees based ranking.

Quality variable	Process variable 1	Process variable 2
Formation Index	1st press shoe nip tilt	Total slice flow
Beta-formation	3rd press bottom felt uhle boxes	1st press top roll saveall
Air permeance	3rd press nip water removal	VacuMaster - vacuum
Air permeance	3rd press nip water removal	LB unit - blades 1-2



(a) Formation Index (b) Beta-formation (c) Air permeance

(d) Formation Index (e) Beta-formation (f) Air permeance

(g) Formation Index (h) Beta-formation (i) Air permeance

(j) Formation Index (k) Beta-formation (l) Air permeance

Fig. 1. Prediction results (a)-(c) on full data set; (d)-(f) on reduced data set, based on the variables obtained by reduced regression trees based ranking. Prediction results (g)-(i) on full data set; (j)-(l) on reduced data set, based on the variables obtained by regression trees based ranking. Dashed line represents true measurements values, solid line corresponds to predicted values, and grayed area stands for $\pm\,\sigma$ interval of predicted values.

and validation (80%), and the test (20%) sets. Once again we keep the mechanical settings balanced in the training and test sets. Then, we repeat the procedure of training a model, validating the model via 10-fold cross validation, and pruning the tree based on the validation results 100 times. However, the algorithm of selecting process variable pairs, which provide the highest impact into the quality variables, differs.

A peculiarity of the used model of regression trees is that they are binary. This implies that a total impact of a variable can be distributed across the tree. To account for this we use a different variable selection algorithm.

For every tree, for each unique variable n in the tree we calculate variable importance scores w_n according to the following formula

$$w_n = \sum_{i=1}^{O_n} \frac{N_{T,n,i}}{N_T} \times \left[\sigma^2_{T,n,i} - \frac{1}{N_{T,n,i}} \times \left(\sigma^2_{L,n,i} N_{L,n,i} + \right.\right.$$
$$\left.\left. + \, \sigma^2_{R,n,i} N_{R,n,i}\right)\right], \qquad (2)$$

where $n = \overline{1:V}$. V is the number of unique variables in the tree, O_n equals number of occurrences of the n-th variable in the tree, $N_{T,n,i}$ and N_T represent number of data points belonging to the i-th entry of the n-th variable and total number of data points used to train the tree, correspondingly. $N_{L,n,i}$ and $N_{R,n,i}$ are numbers of data points that were split from the i-th entry of the n-th variable into the left and right branches, respectively. $\sigma^2_{T,n,i}$ refers to the total variance of data that reached the i-th entry of the n-th variable. $\sigma^2_{L,n,i}$ and $\sigma^2_{R,n,i}$ represent variances of data that reached left and right child nodes of the i-th entry of the n-th variable. Intuitively, this score computes the reduction in the uncertainty of our prediction due to adding rules corresponding to a variable.

After calculating weights of the variables we mark a pair of variables that have highest weight values individually as a candidate pair. The final pairs of process variables are selected among all trees based on the frequency of their occurrence. Thus, we end up with choosing those pairs, which have maximal occurrence frequency among all candidate pairs with highest weights in their trees.

Tables IV, V list the results of this method. Table IV describes the variables obtained on the full data set, and Table V represents results acquired on the reduced data set.

V. Results

This paper presents two approaches to group variable selection in paper industry based on regression trees. Two real data sets were used for testing the methods and comparing the results. As shown in Tables II-V, both methods demonstrate similar results within each data set. However, there are differences in groups of selected variables between the data sets. Figure 1 displays the prediction results using the groups of variables obtained by suggested methods applied to full and reduced data sets. As one can see from Figure 1 prediction for measurements of Beta-formation is less accurate comparing to prediction results for Formation Index and Air permeance, which is similar to outcomes of [7] and [9]. Furthermore, Figure 1 proves that the prediction results for all quality variables are comparable between the data sets.

VI. Conclusions

The importance of variables was analyzed using two methods with the use of two data sets containing real data. To test how well the selected groups of process variables predict quality variables, we estimated the laboratory measurements by applying regression trees to the selected variables groups. There is no significant difference between the results corresponding to different data sets. According to expert opinion, the groups of predictor variables discovered in this research look credible. For the operating personnel, the obtained list of variable pairs gives new information about which combinations of process variables are more valuable in predicting quality variables.

References

[1] T. Ahola, E. Juuso, and K. Leiviskä, “Variable selection and gouping in a paper machine application,” *International Journal of Computers Communications and Control*, vol. 2, no. 2, pp. 111–120, 2007.

[2] A. Isokangas and M. Ruusunen, “Systematic approach for data survey,” in *Proceedings of the International Conference on Informatics in Control, Automation and Robotics. September 14 - 17, 2005, Barcelona, Spain*, 2005, pp. 60–65.

[3] A. Granados, A. Ledezma, G. Gutiérrez, and A. Sanchis, “Reducing the amount of input data in traffic sign classification.” in *3th International Conference Modeling Decisions for Artificial Intelligence, MDAI 2006, Tarragona, Catalonia, Spain, April 3-5, 2006*.

[4] R. Dechter and J. Pearl, “Generalized best-first search strategies and the optimality of a*,” *J. ACM*, vol. 32, no. 3, pp. 505–536, Jul. 1985.

[5] R. Balabin and S. Smirnov, “Variable selection in near-infrared spectroscopy: benchmarking of feature selection methods on biodiesel data.” *Analytica Chimica Acta*, vol. 692, no. 1-2, pp. 63–72, 2011.

[6] P. Patrinos, A. Alexandridis, K. Ninos, and H. Sarimveis, “Variable selection in nonlinear modeling based on rbf networks and evolutionary computation,” *International Journal of Neural Systems*, vol. 20, no. 365, 2010.

[7] P. Nieminen, T. Kärkkäinen, K. Luostarinen, and J. Muhonen, “Neural prediction of product quality based on pilot paper machine process measurements,” in *Adaptive and Natural Computing Algorithms*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6593, pp. 240–249.

[8] I. Jollife, *Principal Component Analysis*, 2nd ed. New York: Springer, 2002.

[9] E. Ivannikova, T. Hamäläinen, and K. Luostarinen, “Information theoretic approach to variable selection in predictive models applied to paper machine data,” in *The 18th IEEE Symposium on Computers and Communications (ISCC’13),July 7-10, 2013, Split, Croatia*, 2013.

[10] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, ser. Statistics/Probability Series. Belmont, California, U.S.A.: Wadsworth Publishing Company, 1984.

[12] G. De’ath and K. E. Fabricius, “Classification and regression trees: a powerful yet simple technique for ecological data analysis,” *Ecology*, vol. 81, p. 31783192, 2010.

[13] A. Floares, A. Jakary, A. Bornstein, and R. Deicken, “Neural networks and classification & regression trees are able to distinguish female with major depression from healhy controls using neuroimaging data,” in *IJCNN*, 2006, pp. 4605–4611.

[14] A. Hecker and T. Kurner, “Application of classification and regression trees for paging traffic prediction in lac planning.” in *VTC Spring*. IEEE, 2007, pp. 874–878.

[15] J.-H. Lee, I.-Y. Kim, and C. M. O’Keefe, “Applicability of regression-tree-based synthetic data methods for business data.” in *ICDM Work-*

PIII

REVEALING COMMUNITY STRUCTURES BY ENSEMBLE CLUSTERING USING GROUP DIFFUSION

by

Elena Ivannikova, Hyunwoo Park, Timo Hämäläinen, and Kichun Lee 2018

Information Fusion, vol. 42, pp. 24-36

Information Fusion 42 (2018) 24–36

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Revealing community structures by ensemble clustering using group diffusion

Elena Ivannikova[a], Hyunwoo Park[b], Timo Hämäläinen[a], Kichun Lee[c,*]

[a] Faculty of Information Technology, University of Jyväskylä, POBox 35 (Agora), Jyväskylä 40014, Finland
[b] Management Sciences at the Fisher College of Business, The Ohio State University, 2100 Neil Ave, Columbus, OH 43210 United States
[c] Industrial Engineering, Hanyang University, Engineering Center #705-1, 222 Wangsimni-ro Seongdong-gu Seoul 133-791, Republic of Korea

ARTICLE INFO

Article history:
Received 10 February 2016
Revised 8 September 2017
Accepted 11 September 2017
Available online 20 September 2017

Keywords:
Clustering
Diffusion
Markov chain
Social network
Community structure

ABSTRACT

We propose an ensemble clustering approach using group diffusion to reveal community structures in data. We represent data points as a directed graph and assume each data point belong to single cluster membership instead of multiple memberships. The method is based on the concept of ensemble group diffusion with a parameter to represent diffusion depth in clustering. The ability to modulate the diffusion-depth parameter by varying it within a certain interval allows for more accurate construction of clusters. Depending on the value of the diffusion-depth parameter, the presented approach can determine very well both local clusters and global structure of data. At the same time, the ability to combine single outcomes of the method results in better cluster segmentation. Due to this property, the proposed method performs well on data sets where other conventional clustering methods fail. We test the method with both simulated and real-world data sets. The results support our theoretical conjectures on improved accuracy compared to other selected methods.

1. Introduction

Interest in the analysis of complex networks has rapidly grown over the past few years. Network models have been used in different areas including economics, biology, social sciences, and computer science, where systems are often represented as graphs. Analyzing network models in practice is a challenging task due to the complexity of the networks, particularly when the underlying community structure is unknown. There are two general approaches to reveal the community structure of networks. The first approach is graph partitioning when the number of clusters is known. The second approach, called community structure detection, is more challenging, as it divides a network into clusters or groups graph nodes when the number of clusters is unknown beforehand. For community structure detection, both identifying clusters and determining the number of clusters must be solved simultaneously.

The detection of community structures in an arbitrary graph is a challenging task. In recent years, several methods have been developed and applied, including min-cut based approaches, clique based approaches, modularity based approaches, clustering approaches, and so forth [1]. These approaches share an essential tool, clustering, in a sense to find good clusters of nodes in a graph that improve a certain criterion. Clustering, indeed, is a universal tool applied in many different fields of data analysis, such as data mining, statistics, marketing, and others [2]. The goal of cluster analysis is to partition data into groups or clusters based on pairwise similarity so that observations inside one cluster are more similar than the ones belonging to different clusters [3]. The dimensionality of the data set has a strong influence on the performance of clustering algorithms. Some methods work well for low-dimensional data, whereas they are unable to find structure in high-dimensional data sets. High-dimensional data pose a number of challenges for researchers and practitioners. First of all, high-dimensional data are more likely to be sparse, which makes it difficult for algorithms to find structures in the data. Moreover, in high-dimensional data sets, points may belong to diverse clusters in different subspaces. Capturing the geometric structure of the manifold from the data, whether low- or high-dimensional, plays an essential role in reliable clustering results. Clustering methods without considering such geometric structures can fail to produce accurate results and find mere local structures in sparse high-dimensional data.

In addition to geometric structures of real-world data, another challenge for clustering in community detection tasks is that the

* Corresponding author.
E-mail addresses: skylee1020@gmail.com, elena.v.ivannikova@student.jyu.fi (E. Ivannikova), park.2706@osu.edu (H. Park), timo.t.hamalainen@jyu.fi (T. Hämäläinen), skylee@hanyang.ac.kr (K. Lee).

https://doi.org/10.1016/j.inffus.2017.09.013
1566-2535/© 2017 Elsevier B.V. All rights reserved.

results cannot be validated as there is no ground truth available for the data sets, as in supervised learning. Furthermore, various clustering methods often generate different and biased structures in a data set due to different optimization criteria they adopt. To overcome these issues in different clustering results, combining multiple partitions can improve the quality of clustering results significantly. In this sense, a collective approach called clustering ensemble aims to provide more robust, stable, and novel solutions by leveraging a consensus of multiple clustering runs. The main goal of clustering ensemble algorithms is to define a clustering solution that maximizes a consensus function and to select a partition generation procedure. Partitioning can be performed using (1) data resampling [4,5], (2) different parameter values or initializations [6,7], and (3) different clustering algorithms such as *k*-means, density-based, graph-partitioning-based, and statistics-based [8].

In this paper, we propose an ensemble clustering algorithm based on the concept of ensemble group diffusion, denoted by Ensemble Group Diffusion, EGD. The proposed method takes into account not only the geometric structure of data using group distances, but also the diffusion of group distances across connectivity scales. Moreover, the presented method is able to produce more robust clustering results by collectively integrating views from individual diffusion scales. The method is also capable of reasonably regulating cluster sizes by an admitted group dependence level. In addition, it nicely handles directed graphs as opposed to other approaches. Further, we present a detailed analysis of the degree of the collective integration and propose a guideline for parameter settings. We demonstrate the efficacy of the proposed method using not only an illustrative simple example, demonstration test cases and simulation studies, but also real-world data sets such as the researcher collaboration network in a healthcare system from the National Institute of Health (NIH) in the U.S., a consumer behavioral pattern captured by the co-purchasing network from Amazon, a leading consumer online shopping place in the U.S. and a gene-expression microarray data sets.

The rest of the paper is organized as follows. Section 2 reviews clustering techniques used in community detection. Section 3 describes preliminary concepts for the proposed algorithm. Section 4 provides theoretical justification for the proposed clustering method. Section 5 compares the performance of the proposed method to popular and state-of-the-art clustering algorithms under different settings and data sets. Section 6 discusses the implications of our development of the algorithm and concludes the paper.

2. Related work

Many clustering algorithms have been proposed in the literature of community structure detection and clustering analysis. Modularity-based methods established by Newman [9] have shown exceptional performance in many cases [10–12]. These methods are nonparametric and are designed to maximize the modularity as an objective function. These methods, however, fail to detect smaller communities in some cases where such granular identification is desirable. It is hard to say whether the detected clusters are indeed single communities or clusters of smaller communities. For example, Fortunato and Barthélemy [13] analyzed modularity-based methods and their applicability in the area of community detection. Their research points out that the modularity function has a resolution limit. Communities that are smaller than a threshold in a certain criterion may not be revealed, even when the whole graph is identified as a single community. In addition, working with pairwise similarity between nodes, modularity-based methods are inherently unable to handle directional relationships commonly observed in reality.

Other clustering methods for community detection also exist. Hierarchical clustering [3], agglomerative or divisive, is another technique commonly used for community detection. Hierarchical clustering [3] first defines a similarity measure between clusters and computes a similarity matrix between vertices of a graph. Among the most critical disadvantages of hierarchical clustering is that the results can be different depending on the similarity measure used, although it is a universal phenomenon in most clustering methods. Besides, agglomerative hierarchical clustering does not scale well, which is crucial for clustering graphs [14]. In essence, approaches based on a predefined number of clusters require an additional important step that involves a decision criterion for the optimal number of clusters. Spectral clustering [3] refers to the group of methods based on eigenvalue decomposition of the similarity matrix or its derivative matrices for clustering data sets. This approach is good at finding non-convex clusters, able to take into account geometric structures of the data [3]. However, it works with similarity matrices, which reflect only bidirectional relationships among the nodes in a graph. The result depends on the number of selected eigenvectors. Along with spectral and density-based methods considering geometric structures of data, Park and Lee [15] proposed a group-dependence clustering approach. This approach is based on the idea of maximizing a measure called group dependence. The central idea of the method is that any two nodes in the graph can be considered as being connected through Markovian transitions. This conceptualization allows for the calculation of ‘dependence distance’ [16] between graph nodes in a certain evolution step, which can adjust the level of connectivity scale in group assignment. Though the method supports the ability to adjust the level of the connectivity scale in clustering, it fails to present a collective view of clusters according to the connectivity scale and was insufficient in coping with directional structures between nodes. Density-based methods detect clusters according to the local density of data points. Based on a density threshold, the points from disconnected regions of high density are assigned to different clusters when the rest are marked as noise. However, such methods, computationally expensive, are suitable only for data defined by a set of coordinates. To overcome these drawbacks, an alternative approach, called clustering by ‘fast search and find of density peaks’ (FSFDP) [17], defines the cluster centers as local density maxima that are relatively distant from any point of higher local density. After that, each remaining point is assigned to the same cluster as its nearest neighbor of higher density.

Attempts to improve the quality of clustering results brought forth developing a number of ensemble clustering approaches during recent years. Zheng et al. used aggregated distance matrices and combined both partitional clustering and hierarchical clustering results [18]. Wang et al. used a Bayesian graphical model to aggregate mixed cluster results and maximized an approximation of the posterior distribution [19]. The clustering approach, proposed in [8] as one of the state-of-the art approaches, addresses the problem of combining multiple partitions of a set of objects using the knowledge-reuse framework [20]. It formulates the cluster ensemble problem by introducing an objective function for combining multiple clustering solutions and by solving the corresponding optimization problem. This way the final consensus solution is obtained without accessing original features. The authors propose the following three consensus functions: cluster-based similarity partitioning algorithm (CSPA) based on a measure of pairwise similarity, HyperGraph Partitioning Algorithm (HGPA) based on approximation of the maximum mutual information objective, and meta-cLustering algorithm (MCLA) based on solving a cluster correspondence problem. The final solution is selected among the three consensus clusterings as the one with the highest average mutual information.

3. Preliminary concepts

In this section, we briefly summarize the concept of group dependence that links data points by a Markov random walk and the concept of a clustering ensemble that combines several division outcomes. Then, in the next section, we propose the concept of ensemble group diffusion to measure a multiscale cohesion level for a group division in an integrative fashion. Ensemble group diffusion gives rise to a new clustering method for community detection, which will be discussed in detail in regards to its characteristics and parameters.

3.1. Group dependence

The concept of group dependence is closely related to the Markov random walk and provides a general foundation for a distance measure between data points that considers the geometrical structure [21]. Group dependence proposed by Park and Lee [15] is a measure that quantifies the goodness-of-division of an undirected graph. In this paper let us suppose that a directed graph of data points (nodes) $x_1, \cdots, x_n \in \mathbb{R}^b$ is given. Denote the set of data points by $\Omega = \{x_1, \ldots, x_n\}$. We view the graph as a Markov chain, assuming the whole chain is ergodic and all transitions follow the Markovian property.

We start with a simple case of bisecting the graph, then providing instructions on how to divide it into more than two groups in the following section. Let $s_i = 1$, a decision variable, if data point i belongs to group 1 and $s_i = -1$ if it belongs to group 2. Observe that the quantity $(s_i s_j + 1)/2$ is 1 if i and j are in the same group and 0 otherwise. Denote the group assignment vector by $\mathbf{s} = [s_1, \ldots, s_n]$. Group dependence is defined as follows:

Definition 1. Group dependence D_t for a given group assignment $\mathbf{s}$ and connectivity scale parameter t is

$$D_t = \sum_{x_i, x_j \in \Omega} \left(Dep(X_0 = j, X_t = i) - 1 \right) \frac{(s_i s_j + 1)}{2},$$

in which $Dep(X_0 = j, X_t = i)$, as dependence, is defined by $\frac{P(X_0=j, X_t=i)}{P(X_0=j)P(X_t=i)}$ and t, as an exogenously given parameter, means the t-step-wide neighborhood evolution in Ω.

Dependence is closely linked to the point-wise mutual information in information theory and the lift measure in association rule learning. Intuitively speaking, dependence captures how x_j in the initial state is inter-dependent with x_i at the tth step: $Dep(X_0 = j, X_t = i) < 1$ means that j and i are negatively dependent; $Dep(X_0 = j, X_t = i) = 1$ means that they are independent; $Dep(X_0 = j, X_t = i) > 1$ means that they are positively dependent. The term, $Dep(X_0 = j, X_t = i) - 1$, represents the degree of relative dependence in comparison to the level of independence as the reference point. Accordingly, group dependence D_t measures the overall coherence of group assignment $\mathbf{s}$ in terms of dependence for the whole data set at the t-step transition. Based on group dependence, we propose another measure of ensemble group diffusion to reflect the multiscale dependence structure in a directed graph. We then look for a good group assignment $\mathbf{s}$ of all n points to maximize the measure.

3.2. Clustering ensemble

As the idea of ensemble group diffusion closely relates to clustering ensembles, we briefly introduce the basic concept of a clustering ensemble. Cluster ensembles basically address the problem of combining multiple base clustering results for the same data set into a final consensus solution. Depending on how to reach a consensus solution, several approaches (such as graph-based, matrix-based, and probabilistic models [22]) exist in the literature. However, the problem formation in cluster ensembles is universal as follows. We start with a base clustering algorithm that generates the group assignment $\mathbf{s}$ of the data points in Ω. We prepare M base clustering results by supplying different parameters to one base algorithm. From them, we obtain M different group assignments $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(M)}$. The results from the M base clustering algorithms can be stacked together to form an overall clustering matrix. Given the overall clustering matrix, the cluster ensemble problem is to combine the M base clustering results for the n data points to generate a consensus clustering, which should be more accurate and stable than the individual base clusterings.

In this paper, we calculate diffusion matrices of dependence for each parameter value of connectivity level t and aggregate the diffusion matrices. Specifically, we similarly start with a directed graph of n data points in Ω with probability matrix P. Having a set of possible parameters, denoted by T, we calculate $[Dep(X_0 = j, X_t = i)]_{i,j=1,\cdots,n}$ for every $t \in T$ and then obtain a cumulative matrix, the ensemble group diffusion, as the sum of the individual group diffusion results:

$$D(\mathbf{s}) = \sum_{t \in T} D_t = \sum_{t \in T} \sum_{x_i, x_j \in \Omega} \left(Dep(X_0 = j, X_t = i) - 1 \right) \frac{(s_i s_j + 1)}{2}. \quad (1)$$

The final clustering is obtained through solving the maximization problem for the cumulative matrix $D(\mathbf{s})$. Combining individual diffusion matrices to ensemble group diffusion can be viewed as a peer regularization. Diffusion matrices obtained with a small t value pull the overall solution towards having smaller clusters. Similarly, diffusion matrices from a large t value shift the optimal solution towards coarser and larger scale representations. The construction of ensemble group diffusion brings stable clustering results under various degrees of resolution and heterogeneous structures in the data set. Thus, it aims to solve the resolution limit issue in which an obvious small-sized community is rarely detected when the whole graph is sufficiently large.

4. Clustering with ensemble group diffusion

We incorporate group dependence and ensemble clustering to present a new approach of ensemble group diffusion that finds the community structure in a directed graph. Given a transition matrix P, we denote the one-step backward transition matrix by P_B, which is calculated from P. Then by backward Markovian transitions, the t-step transition matrix is P_B^t: $P_{B;i,j}^t = P(X_0 = j | X_t = i)$. We observe that if x_i and x_j are close in the geometric structure of the data, the backward transition probability should be large. The posterior transition probability involves a backward Markov chain, representing the probability of the initial state j after reaching state i at the tth step transition as a measure of the difference between the two states i and j in the directed graph.

In particular, the backward transition probability $P_{B;i,j}^t$ should be at least greater than the probability that x_i and x_j are connected by chance among all data points. Thus, the greater $P_{B;i,j}^t$ is among the data points in a cluster, the better the cluster is. Also, a partition of the data set is meaningful when a whole connectivity level by the partition should increase more than that by a random configuration. The quantity $\sum_{i,j} (P_{B;i,j}^t - 1/n)$ should be great for all x_i and x_j pairs in the same cluster, where the threshold probability $1/n$ represents the random probability among n data points. If one seeks a tight configuration, one may use a value larger than $1/n$. In quantifying the connectivity level in detail, we use not only the concept of geometric diffusion, but also modulate the diffusion depth parameter t by varying t in a certain interval. Thus, we define the collective geometric diffusion to be

$$\sum_{t\in T}\sum_{i,j}(P^t_{B;i,j}-1/n),$$

where T is the set of diffusion depth values.

We denote the group assignment vector by $\mathbf{s}=[s_1,\ldots,s_n]$. Let $s_i=1$ if data point i belongs to group 1 and $s_i=-1$ if it belongs to group 2. To obtain a good partition in terms of collective geometric diffusion, we solve the following programming:

$$\text{argmax}_{\mathbf{s}}\sum_{t\in T}\sum_{i,j}(P^t_{B;i,j}-1/n)\frac{(s_is_j+1)}{2}. \tag{2}$$

We show that collective geometric diffusion is in proportion to ensemble group diffusion when all initial states have equal probability as non-informative prior because

$$\begin{aligned}\sum_{t\in T}\sum_{i,j}(P^t_{B;i,j}-1/n) &= 1/n\sum_{t\in T}\sum_{i,j}(P(X_0=j|X_t=i)n-1)\\ &= 1/n\sum_{t\in T}\sum_{i,j}\left(\frac{P(X_0=j,X_t=i)}{P(X_t=i)\frac{1}{n}}-1\right)\\ &= 1/n\sum_{t\in T}\sum_{i,j}\left(\frac{P(X_0=j,X_t=i)}{P(X_t=i)P(X_0=j)}-1\right)\\ &= 1/n\sum_{t\in T}\sum_{i,j}(Dep(X_0=j,X_t=i)-1).\end{aligned}$$

Thus, the programming in (2) is equivalent to maximizing the ensemble group diffusion

$$\text{argmax}_{\mathbf{s}}\sum_{t\in T}\sum_{i,j}(Dep(X_0=j,X_t=i)-1)\frac{(s_is_j+1)}{2}.$$

Note again that the quantity $(s_is_j+1)/2$ is 1 if i and j are in the same group and 0 otherwise. Thus, an optimal clustering scheme is achievable through maximizing the collective geometric diffusion measure by varying the group assignment $\mathbf{s}$ of all n points. To express the level of closeness to a group, the group identity s_i is extended from discrete to continuous with the norm of $\mathbf{s}$ fixed. By the set of diffusion depths T, we can effectively adjust the level of the connectivity scale for which two points are associated. When infinite diffusion steps are taken, for the infinite value of t, the Markov chain converges to the stationary distribution and the collective geometric connectivity becomes trivial. For instance, one can set T to be {1, 2} as a short-range scale or {5, 6, 7, 8} as a mid-range scale. In practice, one could start with T as a long-range scale such as {1, ⋯, 8} and, depending on the result, shrink it, or vice versa (start with T as a short-range scale and expand it). We will see the effect of T by varying it in the experiment section. We express the maximization of the ensemble group diffusion with respect to $\mathbf{s}$ subject to $\|\mathbf{s}\|=1$ as follows:

$$\text{argmax}_{\mathbf{s}}\quad \mathbf{s}^\top\sum_{t\in T}\left(P^t_B-1/n\mathbf{1}\mathbf{1}^\top\right)\mathbf{s}:=\text{argmax}_{\mathbf{s}}\quad \mathbf{s}^\top G\mathbf{s}, \tag{3}$$

where

$$G=\sum_{t\in T}\left(P^t_B-1/n\mathbf{1}\mathbf{1}^\top\right). \tag{4}$$

We numerically find the eigenvalues and eigenvectors of G. The existence of the largest and positive eigenvalue and its eigenvector $\mathbf{s}_1$ implies that the ensemble group diffusion is maximally increased by adjusting a division of the data set on the direction of the corresponding eigenvector $\mathbf{s}_1$. We mention the computational hurdle is computing eigenvalues and eigenvectors of G, and we compare its running times in the experiment section. Moreover, the division of the data points is based on the signs of $\mathbf{s}_1$. In fact, $\mathbf{s}_1$ is a one-dimensional representation of the data points, which can be used for a general purpose such as visualization and classification because the sign and magnitude of $\mathbf{s}_1$ relate to the degree of closeness to one group against the other. We note that the eigenvector associated with the zero eigenvalue represents assigning all data points to just one group. On the contrary, nonexistence of a positive eigenvalue implies any further division of the data yields no gain.

4.1. Detecting more than two groups

The procedure explained so far either divides a graph into two groups or decides not to divide further. It is natural to consider a network with more than two groups latent in its community structure. To obtain more than two clusters, we adopt a standard approach to subsequently divide the groups found [9,15]. We look for a possible division for each group found in the previous step by constructing a new backward transition matrix $P_{B|g}$ for a detected group g as a subset of the data set: $P_{B|g}$ with size $|g|\times|g|$, defined by

$$P^t_{B|g;i,j}=\{P(X_0=j|X_t=i)|\forall i,j\in g\}.$$

Following the same procedure based on $P_{B|g}$, the solution of $\text{argmax}_{\|\mathbf{s}\|=1}G^{(g)}$ in (3) enables us to decide whether a division is possible or not.

It is important to note that the new group configuration with a new division found does not always result in an increase in the ensemble group diffusion of the whole graph because the new similarity matrix $P_{B|g}$ reflects only a part of the whole data set without considering connections to the nodes belonging to other groups. Hence, among the possible divisions, we look for only the division which causes the ensemble group diffusion in (3) for the whole data set to increase most when the new division is applied.

Furthermore, we require that the increase is at least by certain margin. Specifically, for the purpose of regularizing the solution, we introduce a dependence gain parameter $\delta_d\in[0,1]$. This parameter is used for calculating the minimal dependence gain value required to split a cluster into two sub-clusters and is defined as

$$\Delta_d=\delta_d\sum_{g_{i,j}>0}G,$$

where $G=[g_{i,j}]$ is defined in (4). For setting the value of δ_d in practice, we recommend starting with quite a small value, close to zero, and increasing it depending on the results. One can verify that $\sum_{g_{i,j}>0}G$ is the upper bound for cumulative collective geometric diffusion. Thus, the division into sub-clusters proceeds if the difference between the collective geometric diffusion values corresponding to the group configurations before and after the division is higher than the dependence gain:

$$D(\mathbf{s}')-D(\mathbf{s})>n\Delta_d,$$

where D is defined in (1), $\mathbf{s}$ and $\mathbf{s}'$ denote the group configurations before and after the division into sub-clusters, respectively, and n stands for the size of the set of data points Ω. The dependence gain parameter prevents the algorithm from identifying clusters that are too small or not clear enough, which allows controlling the desired level of clarity in finding clusters. In summary, we stop dividing the group when we find no positive eigenvalues from group ensemble matrix $G^{(g)}$ or the dependence gain fail to exceed $n\Delta_d$.

4.2. Illustrative examples

This section demonstrates the performance and steps of the proposed algorithm. For illustration, we construct a simulated similarity matrix, which is generated as follows. We randomly generate 500 points in two dimensional space where every point belongs to one of three clusters. One cluster is generated from a

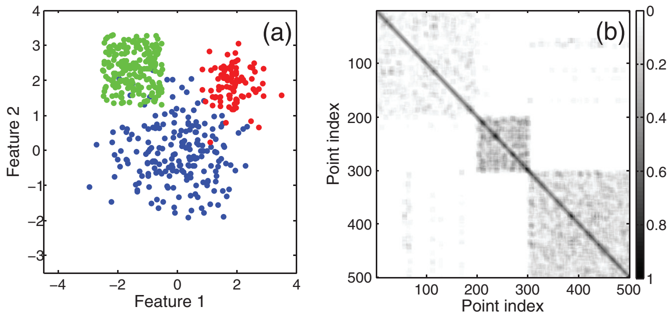


Fig. 1. (a) The original data set and true cluster division from illustrative example, (b) similarity matrix corresponding to the data in (a).

Gaussian distribution with identity covariance matrix **I** and contains 200 points. The second cluster is formed by a Gaussian distribution with covariance matrix 0.25 × **I** and consists of 100 points. It is shifted from the center of the first cluster by approximately 2.7 units. The third cluster is constructed from a uniform distribution in the interval (0, 2). It consists of 200 points and is shifted from the center of the first cluster by approximately 2.8 units. The obtained data is illustrated in Fig. 1(a), where clusters are colored differently. The data set has quite a visible underlying community structure, although cluster membership for some of the points on the border is not clear. We construct a distance matrix using the Euclidean distance. The distance between two points x and y is calculated as $\|\mathbf{x}-\mathbf{y}\| = \sqrt{\sum_i (x_i - y_i)^2}$, where i denotes a dimension. The similarity matrix shown in Fig. 1(b) is formed from the distance matrix using the general form of a Gaussian radial basis function

$$h(x, y) = \exp(-\|x-y\|^2/\sigma^2), \tag{5}$$

where $\sigma > 0$. In this example, considering the units of $x_i - y_i$, we empirically set the parameter σ to 0.2 and δ_d to 0.12. We notice that three standard deviations of $x_i - y_i$, $3\sigma_{x_i - y_i}$, is 0.212 and the choice of δ_d from the interval [0.05, 0.12] brings no change in the clustering result. Therefore, we choose the upper limit of this interval $\delta_d = 0.12$.

We apply the proposed method (EGD) clustering to the obtained similarity matrix. We consider the diffusion depth parameter values $T = \{3\}$, $T = \{8\}$ and their combination $T = \{3, 8\}$. Fig. 2(a)–(c) shows clustering results with EGD for the data and parameters described above. Every cluster is marked in its own color. The clusters corresponding to original data clusters are displayed in similar colors. The algorithm with $T = \{3\}$ fails to find the underlying data structure and assigns nearly all points to one cluster (see Fig. 2(a)). For $T = \{8\}$, the method successfully determines one cluster marked green (see Fig. 2(b)), but shuffles the two remaining clusters. For $T = \{3, 8\}$, the proposed EGD approach discovers all three clusters, except for a few elements near the border and a small group of points treated as a separate cluster. We note that the border between clusters in this data set is not obvious and the problem of clustering such points belonging to the border will be addressed further in Section 6. Apparently, the collective nature of the proposed method leverages the benefits of runs with a single t (see Fig. 2(c)).

Furthermore, using the data set with ground-truth, we show the effectiveness of the EGD algorithm by providing clustering results under various settings of the parameter T. Figs. 2(d), (e), and (f) display clustering results for $T = \{2\}$, $T = \{7\}$ and their combination $T = \{2, 7\}$, respectively. The algorithm with $T = \{2\}$ assigns nearly all points to one cluster (see Fig. 2(d)). For $T = \{7\}$, the method determines two clusters marked green and blue, but mixes points in the third cluster (see Fig. 2(e)). For $T = \{2, 7\}$, the EGD algorithm discovers all three clusters, except for a few elements near the border and a small group of points treated as a separate cluster. Clustering results for $T = \{1\}$, $T = \{9\}$ and their combination $T = \{1, 9\}$, displayed in Figs. 2(h), (i), and (j), are similar to the ones obtained for $T = \{3\}$, $T = \{8\}$, and $T = \{3, 8\}$, and consistently show the benefit of collective diffusion depths.

To demonstrate the functioning of the ensemble algorithm, we display how the data set is split by the eigenvectors of the ensemble group diffusion matrix G in (4) in every iteration. We consider the case when $T = \{3, 8\}$. The first eigenvector corresponding to the first iteration of the algorithm splits the data set into two clusters, green and black (see Fig. 3(a)-(b)). At the second iteration, its own first eigenvector splits the green cluster into two parts, green and blue (see Fig. 3(c)-(d)). Last, the first eigenvector corresponding to the third iteration separates the green cluster from the previous step into two subclusters, marked green and red (see Fig. 3(e)-(f)). In Fig. 3 values of the eigenvectors and the corresponding points on the scatter plots are displayed in the same color. In addition, we add Figs. 10 and 11 in Supplementary Materials to show how the data set is split by the eigenvectors of the matrix G in every iteration for $T = \{2, 7\}$ and $T = \{1, 9\}$, respectively. The results demonstrate that the algorithm is stable under various parameter settings and provides reasonable separation into groups. Then, we set $\delta_d = 0$ to promote cluster splits. We run EGD by varying t from small to bigger values to show how cluster structure evolves by changing t values. Fig. 4 demonstrates the effect of increasing t on clustering results, evolution from local to global structure.

For comparison, we provide the results of the modularity, spectral, and hierarchical clustering methods used further in this work (see Fig. 5). As parameter values for spectral and hierarchical methods we provide true ($k = 3$) and wrong numbers of clusters ($k = 2, 4$). Hierarchical clustering method places all data points mainly in one cluster for all tested parameter values (see Fig. 5(a)–(c)). Spectral clustering correctly determines one cluster for $k = 5$ but shuffles points belonging to the other two clusters. However, for $k = 3$, spectral clustering performs relatively well, which is quite natural in that the true number of clusters was provided in this case. The results of the spectral clustering approach can be seen in Fig. 5(d)–(f). The modularity approach fails for this data set as it discovers too many clusters (see Fig. 5(g)).

Next, we applied the EGD algorithm to the test cases in which underlying manifold structures exist, as presented in Fig. 6. Fig. 6(a)–(c) refer to artificial data sets representing classes of different shapes [23]. Fig. 6(d) displays the test case which the FLAME

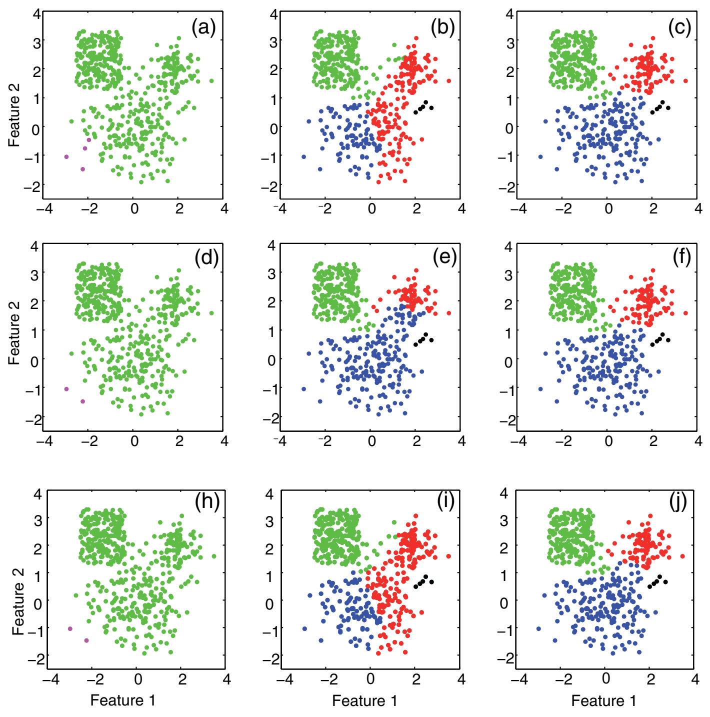


Fig. 2. EGD clustering results for illustrative example: (a) $T = \{3\}$, (b) $T = \{8\}$, (c) $T = \{3, 8\}$, (d) $T = \{2\}$, (e) $T = \{7\}$, (f) $T = \{2, 7\}$, (h) $T = \{1\}$, (i) $T = \{9\}$, (j) $T = \{1, 9\}$. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

approach [24] used as a challenging test case. For computing distance matrices we adopted Manhattan distance measure for the two spirals data set displayed in Fig. 6(a) and standardized Euclidean distance measure for the remaining test cases. Similarity matrices were calculated from distance matrices using Gaussian radial basis function as in (5). EGD successfully determined true clusters for the test cases as shown in Figs. 6(a)–(c). In particular, the results for the test case (d) are comparable to the original method. Note that unlike original method, EGD assigned two outlier points displayed in red to a separate cluster that looks natural in this case and shows a potential ability to reveal a community that is small in scale.

The illustrative and demo examples show that the proposed EGD clustering approach can determine the underlying geometry of the data, even for those data sets where some of the common clustering methods fail. We attribute it to the property of ensemble group diffusion that inherently combines individual outcomes to result in better cluster segmentation.

5. Experimental and empirical results

5.1. Benchmark methods

We compare the EGD clustering with other well-known methods frequently used for community structure detection. Among those methods are agglomerative hierarchical clustering, spectral clustering, modularity clustering [9], density-based clustering and a knowledge reuse framework-based (KRF) clustering ensemble approach proposed in [8]. We apply the KRF approach to the results obtained by Metis [25] and graph partitioning (GP) [26] algorithms by varying the number of clusters.

As a density-based method we refer to clustering by 'fast search and find of density peaks' (FSFDP) by Rodriguez and Laio [17]. We use a Matlab implementation of FSFDP as was expounded in [27]. In order to define cluster centers the original method adopted a manual setting by supervised analysis of a decision graph that displays local density ρ_i versus distance δ_i from points of higher density for each data point *i*. Then one finds a rectangular region where both δ_i and ρ_i are high [27]. As the procedure was quite inefficient and deteriorated in the multiple data sets used, we designed a heuristic for automatic selection of cluster centers by binning the data points into *k* equally spaced intervals along the axes and marking points with maximal δ in each bin as cluster centers.

We employ both simulated and real-life data sets to compare performance of the clustering algorithms, including the proposed EGD method in this paper. In order to evaluate the performance of these algorithms, we need to have "true clustering labels" for each data set. The simulated data set clearly has one, as we simulate the data set from a predefined correlation matrix structure. For other real-world data sets, we deliberately chose empirical settings where we can define such true clustering for all nodes.

5.2. Performance evaluation

Given true clustering labels, we measure the performance of the clustering results using two approaches: Rand measure [28] and normalized mutual information (NMI) [8]. The Rand measure is based on the dyad-level accuracy of clustering results, counting the

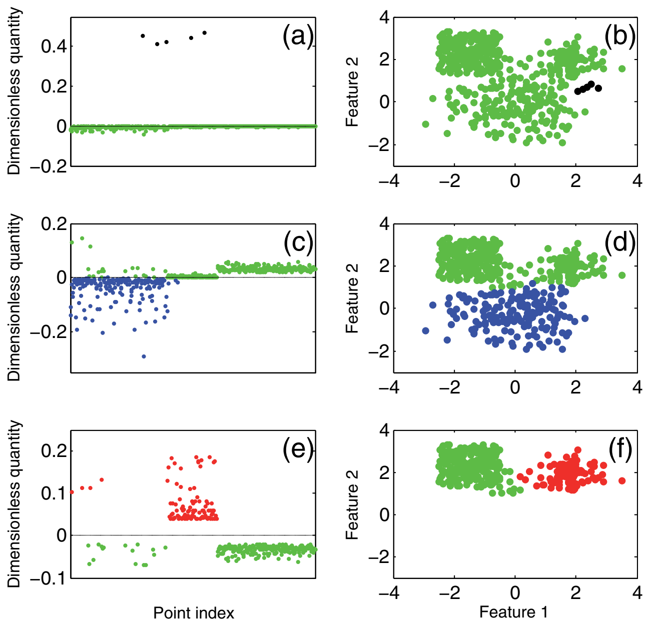


Fig. 3. EGD clustering steps in illustrative example for $T = \{3, 8\}$. (a) The first eigenvector of similarity matrix at the first iteration, (b) the first division, (c) the first eigenvector of similarity matrix at the second iteration, (d) the second division, (e) the first eigenvector of similarity matrix at the third iteration, (f) the third division. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

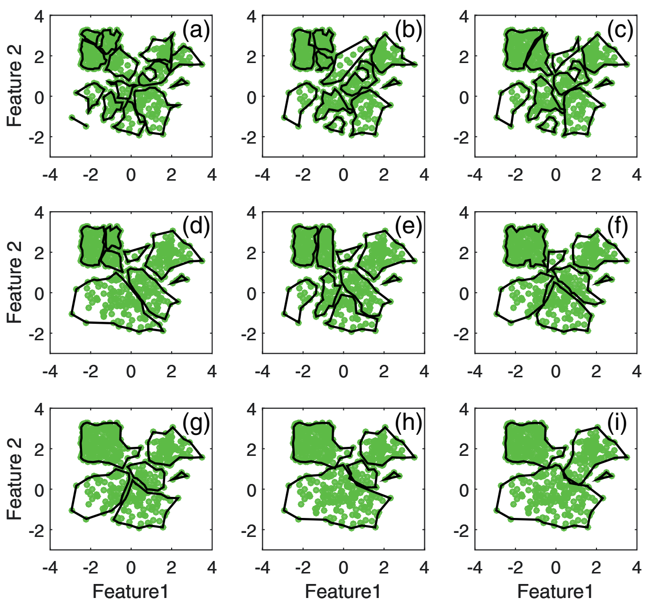


Fig. 4. Evolving cluster structure by changing t (number of defined clusters is shown in brackets): (a) $T = \{1\}$ (14), (b) $T = \{5\}$ (11), (c) $T = \{9\}$ (10), (d) $T = \{12\}$ (8), (e) $T = \{15\}$ (9), (f) $T = \{25\}$ (7), (g) $T = \{29\}$ (6), (h) $T = \{35\}$ (5), (i) $T = \{40\}$ (4).

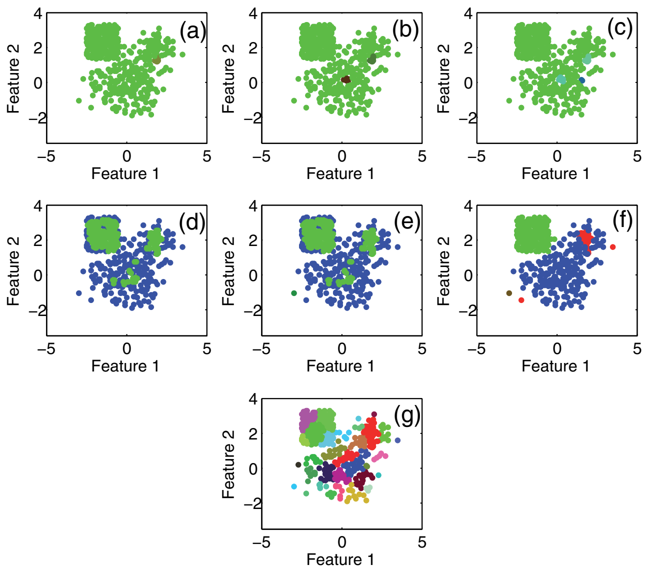


Fig. 5. Clustering results for illustrative example: (a) hierarchical clustering ($k = 2$), (b) hierarchical clustering ($k = 3$), (c) hierarchical clustering ($k = 4$), (d) spectral clustering ($k = 2$), (e) spectral clustering ($k = 3$), (f) spectral clustering ($k = 4$), (g) modularity clustering.

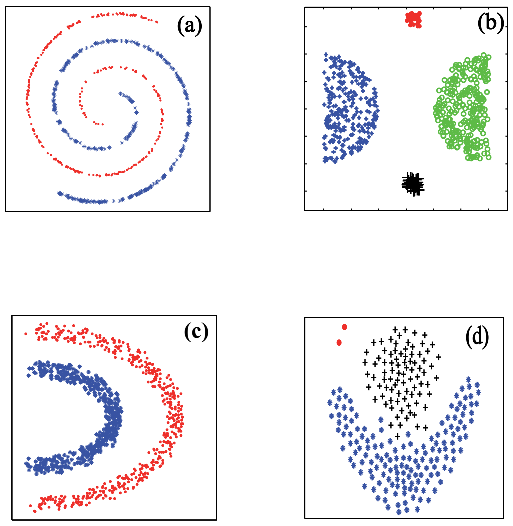


Fig. 6. EGD clustering results for the data sets: (a) two spirals, (b) outlier, (c) half kernel, (d) FLAME.

number of pairs in which an algorithm's clustering result and the true clustering specification agree. In essence, it combines the ratio of correctly clustered pairs (CC) and the ratio of correctly separated pairs (CS). CC and CS quantify the performance of clustering algorithms in terms of what percentage of pairs are correctly clustered or separated given the true clustering results. They represent two extremes of a clustering algorithm's performance. If an algorithm tends to cluster aggressively by putting too many nodes into the same cluster, it will score high in CC but low in CS, and vice versa. Thus, a desirable clustering algorithm should score high in the Rand measure, balancing CC and CS.

These three measures are computed as follows:

$$CC = \frac{\sum_{i,j} 1_{\{x_i=x_j\}} 1_{\{y_i=y_j\}}}{\sum_{i,j} 1_{\{x_i=x_j\}} 1_{\{y_i=y_j\}} + \sum_{i,j} 1_{\{x_i\neq x_j\}} 1_{\{y_i=y_j\}}},$$

$$CS = \frac{\sum_{i,j} 1_{\{x_i\neq x_j\}} 1_{\{y_i\neq y_j\}}}{\sum_{i,j} 1_{\{x_i=x_j\}} 1_{\{y_i\neq y_j\}} + \sum_{i,j} 1_{\{x_i\neq x_j\}} 1_{\{y_i\neq y_j\}}},$$

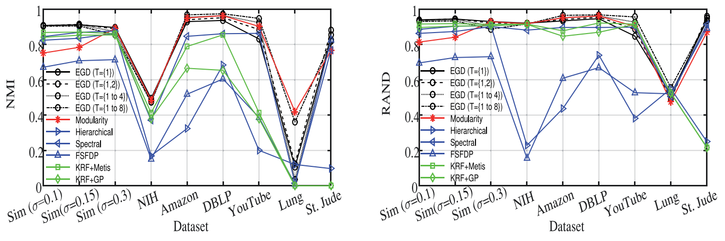
$$Rand = \frac{\sum_{i,j} 1_{\{x_i=x_j\}} 1_{\{y_i=y_j\}} + \sum_{i,j} 1_{\{x_i\neq x_j\}} 1_{\{y_i\neq y_j\}}}{\sum_{i\neq j} 1},$$

where $X = \{x_i\}, i = 1, \cdots, n$ is the clustering result under evaluation and $Y = \{y_i\}, i = 1, \cdots, n$ is the true cluster labels. n represents the number of nodes in the graph.

On the other hand, we employ another measure, NMI, to capture similarity between the true and test clustering results in a holistic way. Mutual information is a concept from information theory and increases as two input sequences are similar to each other. The normalized variant that we use in this paper scales it into the range between zero and one. The normalization process is similar to that of the Pearson correlation coefficient. In this case, the information-theoretic entropy serves as a normalization factor. The information entropy measures how random each input sequence is. Following Strehl and Ghosh [8], NMI is calculated as follows:

$$\mathrm{NMI}(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}},$$

where $I(X, Y)$ denotes mutual information between X and Y, and $H(X)$ and $H(Y)$ denote the information entropy of X and Y, respectively.



(a) (b)

Fig. 7. Performance summary of the proposed method and other methods in terms of (a) NMI, (b) RAND.

We show the performance summary of the proposed method and other methods in terms of NMI and RND across 9 data sets in Fig. 7. We give detailed description of the data sets and performance comparisons in the following sections.

5.3. Simulation tests

Our evaluation starts with the tests on synthetically generated data sets. We use a simulated correlation matrix with four evident groups, which represents a graph of 12 nodes. The group structure imposed into the correlation matrix is {1,2}, {3,4,5,6}, {7,8,9}, {10,11,12}. Within-group true correlation coefficients are 0.9, 0.7, 0.6, and 0.8. Nodes in groups 1 and 2 are positively correlated by 0.2 and those in groups 3 and 4 are negatively correlated by −0.4. All other inter-group correlations are set to zero. Each node represents a random variable and the edges of the graph are Pearson sample correlation coefficients $r_{i,j}$ in the range from −1 to 1. The distance between two points i and j is calculated as $\|x_i - x_j\| = 1/(r_{i,j} + 1)$. For calculating the similarity matrix W, a general form of the Gaussian radial basis function in Eq. (5) was used. More detailed information about the data can be found in [15,29].

We evaluate each clustering method as an average over 10,000 replications. For each replication, we build a sample correlation matrix from random realizations from the true correlation matrix. Following Stone and Ayroles [29], we extract nine observations from the true correlation matrix using a multivariate normal distribution. In order to see the effects of the width parameter σ, we vary σ from 0.1 to 0.15 and 0.3.

We tested EGD by varying the diffusion depth parameter set as $T = \{1\}$, $\{1,2\}$, $\{1,2,3,4\}$, and $\{1,2,3,4,5,6,7,8\}$ from short-range to long-range scales. Our benchmark methods include agglomerative hierarchical clustering, spectral clustering, modularity clustering, density-based clustering and KRF applied to the results of Metis and GP. We informed benchmark methods other than Modularity about the number of clusters with both the true value ($k = 4$) and misleading values ($k = 3, 5$). Table 4 in Supplementary Materials shows the results of simulation experiments using the nine clustering methods. Boldface values denote the highest number in each column. EGD outperforms other methods for all values of σ. The method demonstrates more accurate results for relatively low values of $\sigma = 0.1, 0.15$ and low values of the parameter δ_d, with the highest NMI and Rand scores of 0.91 and 0.94, correspondingly for both σ values.

To verify the performance differences between the proposed method and the other methods, we applied post statistical analysis using repeated measures ANOVA. The tests showed that the proposed method outperformed the other methods in the simulation experiments with 95% confidence levels. Please refer to Supplementary Materials (Table 2) to see the p-values of the tests.

5.4. Co-PI network from the NIH research funding data

To benchmark the performance of the proposed method in the real-world context, we constructed a principal-investigator (PI) network from the funding data of the National Institute of Health (NIH) of the United States. The NIH is a collective body of 27 institutes and centers (ICs), disbursing $25-30B every year for biomedical research. This accounts for a significant portion of the total biomedical research funding of the U.S. and the NIH is the single largest public entity in the picture. As a public agency, the NIH keeps track of detailed funding information for each grant including grant application abstract, activity type, amount of grant, list of co-PIs, and institution of the head PI.

For each grant, one or more researchers are in charge of carrying out the proposed research project. Among them, one person is designated as head PI (or contact PI), who is meant to be the primary corresponding agent for the project. Each project record also contains the head PI's associated institution (university, research institute, or private company) and its address. Although most grants are executed by a single PI, a few projects are led by multiple PIs, in which case the project is run by a head PI and co-PIs.

We focus on the projects having multiple co-PIs to construct the collaboration network of researchers in biomedical research. By counting the number of co-occurrences of PIs, we obtain the weighted undirected graph of the co-PI network.

In order to test the clustering algorithms, we need not only a network but also true labels of the nodes. We prepared the true labels based on the location of the affiliated institution of a PI. In essence, we infer whether PIs are co-located from the collaboration network structure among the co-PIs. Reasoning behind this inference is that researchers in the same geographical region are more likely to collaborate on research project supported by NIH funding.

We first collected all grant data between 2000 and 2012 from NIH's data retrieval interface called ExPORTER. The NIH publishes funding records not only for its 27 ICs but also for some other related agencies. Then, a small number of research grants are awarded to non-US institutions. Last, some large projects are broken into subproject records occasionally. In such cases, we only consider the ultimate parent project record. Since our focus is on the NIH's U.S. funding records, we remove non-U.S. projects from our sample. After filtering out non-NIH, non-US grants, and subproject records, we are left with 707,496 grants. 14,093 projects among them have more than one PI and the number of unique PIs is 11,999. As institution information is only available for the head PI, we removed PIs for which we cannot identify the institution, and 9769 PIs remain. The collaboration network is extremely sparse because of a myriad of isolated cliques of two or three PIs. We extracted the connected components of size greater than or equal to 10 from the entire landscape. At last, we are left with 993 PIs from 217 institutions in 44 states. Assuming that investigators affiliated with institutions that are geographically close to each other have a higher chance to collaborate as co-PIs, we use the state as a true clustering label for each PI based on the location of their institution.

Table 5 in Supplementary Materials shows the performance comparison with the NIH data set. The EGD clustering with $\delta_d = 0$ and $T = \{1\}$ outperforms all other configurations and algorithms, including modularity clustering. We observe a high Rand measure and low NMI consistently across all methods. Low CC and high CS scores explain why Rand measure is higher than NMI. This implies that the co-PI network is highly fragmented and it has a fairly low chance that two PIs are associated with the institutions in the same state. When algorithms place nodes into the same cluster, it is more likely to be wrong than when they separate out nodes into different clusters. Thus, under this sparse clustering structure, we see that NMI is a more robust measure than the Rand measure, although these two measures were close to each other in the simulation study in the previous section. Last, the hierarchical clustering and FSFDP clustering scores are low in both Rand and NMI, which suggests that the methods clearly failed to correctly identify the latent community structure. Spectral clustering, Metis, GP, and KRF produced better results, but still fell short of the results from modularity clustering and the EGD clustering.

5.5. Social network data with ground truth membership records

Social networks have gained a significant number of users over the past decade. Various social network services operate in the web with a different focus, such as for friendship or professional career networks. This trend led to an explosion of availability of social network data that could be used for academic research. Indeed, various fields such as marketing and psychology have used data sets from real world social network services to address a specific research question. Clustering algorithm development is one of the fields that can immediately benefit from using these social network data sets. One hurdle that prevents one from doing such research is that raw social network data usually does not provide ground truth membership of the nodes. In order to test clustering algorithm performance, we need a true clustering that can be compared against as a benchmark. The notion of ground truth membership depends on how you frame the clustering task. For instance, suppose that we are interested in clustering people in a professional career social network. Depending on our research interest, community structure may be defined by age group or the industry that they are working in. In this case, both age group and industry code can serve as the ground truth membership label for each person in the network.

This section is devoted to the analysis using ground truth networks provided by Yang and Leskovec [30], who constructed a large set of networks with explicit ground truth community structure from a number of different domains. We apply the EGD algorithm to the Stanford Network Analysis Project (SNAP) data consisting of three data sets and compare its performance with the benchmark methods in the same way as before. SNAP data used in this section are in fact undirected graphs with binary edge weights describing three well known real world networks.

The first data set is Amazon's product co-purchasing network. The data set is constructed based on the feature which lists corresponding products (goods) under the tag “Customers Who Bought This Item Also Bought” [30]. The ground truth community is constructed in a way that all its members share a common purpose. Amazon-defined product categories (e.g., electronics, beauty & health, or clothing) serve as the ground-truth communities. The second data set is from DBLP, which is a widely known bibliography repository archiving archiving publication records particularly focusing on the field of computer science. Yang and Leskovec [30] extracted authors' collaboration network from publication data. The authors are connected if they have a joint publication. The publication venues serve as ground-truth communities for the authors. Last, the third data set comes from YouTube, an online video sharing community. It acts as a social network where users can form friendships, create own groups, and join other groups. Such group membership provides ground-truth communities of the users.

In the original data set, Yang and Leskovec [30] provided the list of the top 5,000 largest communities along with network data (i.e., nodes and edges). Since the size of the networks is too large, we first need to reduce the data to check how the clustering algorithms work with smaller data sets. We preprocessed the data in a way that we randomly select top 10 mutually exclusive communities. This guarantees that each node belongs to only a single community, which clears the ambiguity concern of multiple membership. Second, in order to lift the computational burden, we reject a sample containing more than 300 nodes. Last, we randomly choose 100 samples to construct the final set of samples. The network and community statistics averaged over 100 samples are as follows. The average number of nodes and edges in three data sets (Amazon, DBLP, YouTube) are $(132, 107, 103)$ and $(387, 314, 171)$, respectively. All data sets have similar number of nodes, but samples from YouTube are much more sparse networks, as they have about half the number of edges compared to the other two data sets. The average clustering coefficients are $(0.69, 0.88, 0.30)$; DBLP exhibits the highest level of clustering coefficients. In sum, these three sets of samples have different network-level characteristics, which allows us to examine the sensitivity of algorithm performance by comparing the algorithms in these three different settings.

Table 6 in Supplementary Materials shows the performance comparison among the nine clustering algorithms with various configurations. In these results, the EGD method predominantly outperformed all other benchmark algorithms. $\delta_d = 0$ and 0.001 produce the best outcome and a larger set of t led to a better outcome than the smaller set, such as $T = \{1\}$. The overall accuracy scores measured in NMI are in the descending order of DBLP (95.74%), Amazon (94.05%), and YouTube (88.81%), which suggests that higher average clustering coefficient is associated with more accurate clustering outcomes. The statistical testing for the three data sets showed that the proposed method outperformed the other methods in the SNAP experiments with 95% confidence levels except for comparisons with spectral clustering. Notice that for DBLP and YouTube we informed spectral clustering of the correct number of clusters. To see the p-values of the tests, refer to Supplementary Materials (Table 3).

5.6. Amazon co-purchasing relationships

Since our proposed method and the dependence clustering works on the adjacency matrix of the network, it is not limited to undirected graphs. Rather, we surmise that our method may work better on directed graphs compared to other favored choices of clustering methods. We construct a set of directed graph samples from Amazon's co-purchasing relationship between products. If product i is purchased together with product j frequently, we denote the relationship as a directed edge from i to j. Note that this relationship is not necessarily reflexive because the absolute level of demand for the two products may starkly differ. This data set is also compiled by SNAP [31]. SNAP collected the co-purchasing network data at multiple points in time. The version we used to construct our samples was collected by SNAP on June 1, 2003. The original population data set consists of 403,394 nodes and 3,387,388 directed edges.

Product co-purchasing networks can serve our purpose of testing the clustering algorithms only if we also have true labels of all nodes. SNAP also provides the metadata for each node such as the product name, product group, and optional subcategories that the product belongs to. SNAP collected the metadata in summer 2006, approximately three years after the co-purchasing network was collected. However, we argue that this gap in data collection time does not affect our samples and results in a significant way because product group and subcategories do not change frequently over time. Most of the nodes fall into one of four product groups: books, music CDs, videos, and DVDs. We decide to select one product group and choose books only for our final samples for three reasons. First, we narrow down to a single product group because we suspect co-purchasing links exist extremely sparsely across different product groups. Second, books are highly standardized products and have a well-defined classification scheme based on the topical subject. Third, books represent more than 70% of the original SNAP data set, thus, choosing books does not undermine the representativeness of our samples. In order to uniquely assign each book to a single true label, we pick the most frequent subject category for a book when the book is tagged with multiple subject areas.

With these settings in place, it is impractical for us to run various clustering algorithms on the full data set. We thus create samples from the full data set with which we test and compare the performance of different clustering algorithms within a reasonable amount of time. The detailed sampling steps are as follows. First, we choose a random book and the randomly chosen book then becomes the only member of the seed set. Second, for each book in the seed set, we look up books frequently co-purchased with the focal book. Third, we add the co-purchased books to the seed set. Fourth, we repeat Step 2 and 3 until the size of the seed set reaches the previously defined threshold. We set the threshold at 100 for our sampling process, so all of our sampled networks have at least 100 nodes. Last, we extract all directed edges between the nodes in the final seed set. In essence, this sampling process generates multiple layers of egonetworks superposed to each other. The adjacency matrix resulting from the sampling process is binary and asymmetric. We create 10 sample networks and labels using this sampling process. The performance metrics are averaged across the 10 samples when reported in the results section.

Each sampled network, on average, contains 165.3 nodes and 774.8 edges, which results in an average network density of 0.03132. 34% of the directed edges are reciprocal, which means that the two nodes have a bidirectional relationship in such cases. The frequently co-purchased relationship is not reflexive by itself, but a significant portion of the relationship in our samples is indeed bidirectional, largely because of our sampling process relying on egonetworks. Still, more than 60% of the relationships are unidirectional. The average number of subject categories for each sampled network is 22.3. One may suspect that most of the nodes in a sampled network belong to a single category also because of our reliance on egonetworks for sampling. However, category membership turns out to be quite evenly distributed. The most frequent category accounts for only 16% of the nodes in a network and the average Herfindahl-Hirschman Index, representing the concentration of proportions, is 0.0873, which is not particularly high.

Table 7 in Supplementary Materials shows the performance comparison between the proposed EGD and modularity clustering methods. After the previous tests we decided to narrow down the comparison analysis to these two methods, as they show the most stable results and both methods do not require a parameter specifying the number of clusters. EGD with low values of t and δ_d outperforms modularity clustering in terms of NMI. Moreover, our method is able to inherently handle the directed relationship by a transition matrix, whereas the modularity approach forces the directed relationship to be symmetric. Thus, compared to modularity clustering, our approach can better handle problems where data sets with inherent directed nature are involved.

Table 1
Description of the data sets.

Data set	# classes	# samples	# features	Distance/Similarity
Lung cancer	4	197	1000	Euclidean/ Gaussian ($\sigma = 0.1$)
St. Jude leukemia	6	248	985	Standardized Euclidean/ $\lvert D - \max(D)\rvert$[*]

[*] D refers to distance matrix

5.7. Gene-expression data

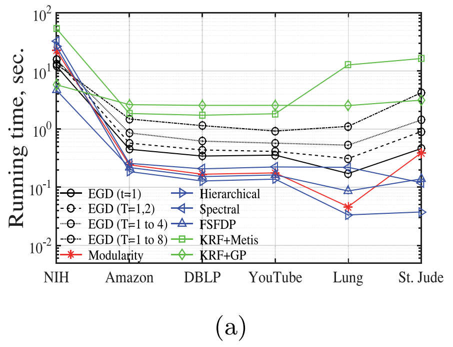
In this section we consider two high-dimensional data sets as regards to gene-expression profiles. One is the lung cancer data set [32] including four known classes of speciments: 139 adenocarcinomas (AD), 21 squamous cell carcinomas (SQ), 20 pulmonary carcinoids (COID), and 17 normal lung (NL). The other is St. Jude leukemia data set [33] that contains samples from pediatric acute lymphoblastic leukemia patients. The data set includes six leukemia subtypes: 43 T-lineage (T-ALL), 27 E2A-PBX1, 15 BCR-ABL, 79 TEL-AML1, 20 MLL rearrangements, and 64 hyperdiploid karyotype (i.e., $>$ 50 chromosomes). More detailed description of the data sets can be found in Table 1.

For both data sets similarity matrices are obtained from the distance matrices using the measures described in Table 1. For the lung cancer data set the test results show that EGD outperforms other approaches demonstrating the highest NMI and Rand measure scores, as shown in Table 8 (Supplementary Materials). Note that though Modularity clustering method performs better in terms of NMI, it assigns every point to a separate cluster, which is hardly practical. Moreover, hierarchical clustering that provides the highest Rand measure score allocates nearly all the points to a single cluster with an exception of only a few samples. The same happens to FSFDP and KRF. Spectral clustering neither succeeds.

For the St. Jude leukemia data set EGD performs best with the highest NMI value of 0.88, as shown in Table 9 (Supplementary Materials). The highest Rand measure score of 0.96 is provided by spectral clustering. However, compared to spectral clustering the loss of EGD in Rand measure is not significant. Hierarchical and KRF approaches fail by assigning the majority of the points to a single cluster which is verified by significantly low values of CS and high values of CC.

5.8. Running times

Finally, the running time for the methods and the data sets used in this work are displayed in Fig. 8. The experiments were conducted on a system with the following characteristics: 64-bit Windows 10 operating system, Intel(R) Core(TM) i5-3317U CPU 1.70 GHz, 8GB RAM, and MATLAB (R2014b). MATLAB implementations of the KRF approach, Metis, and GP used in the experiments are available at [34,35]. For the methods which require the number of clusters as a parameter, only iterations when the true parameter values are provided were considered during the running-time evaluation. Fig. 8(a) displays running times on the logarithmic scale grouped by the data sets and averaged over the data sets, correspondingly. Fig. 8(b) shows running times in seconds for each method averaged over the data sets. The proposed EGD demonstrated good performance and proved to be the most efficient for the majority of the data sets among the ensemble methods used for comparison in this work. In addition, we show running times according to the length of input data in Fig. 9. The data set consists



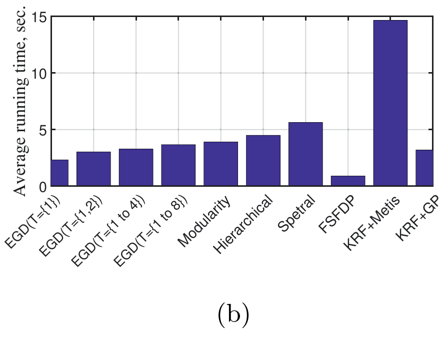


Fig. 8. Running times of the methods (a) grouped by the data sets (logarithmic scale), (b) averaged over the data sets.

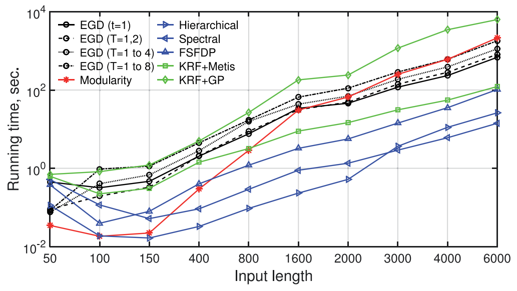


Fig. 9. Running times of the methods against data input length (logarithmic scale).

of randomly generated points in two dimensional space where every point belongs to one of three clusters (n points in a cluster), similarly to the example from Fig. 1(a). In the tests, n varies in the interval from 50 to 6000. The experiments were conducted on a system with the following characteristics: 1TB of RAM, 64 CPU cores (8 cores Intel(R) Xeon(R) E7-8837 2.67 GHz per CPU), and MATLAB (R2017a). The proposed EGD showed consistent performance among the tested methods. One can see that EGD is comparable to Modularity, outperforming it for large input lengths ($n > 2000$) and KRF+GP for all the values of T.

6. Discussions and conclusion

Cluster ensemble algorithms recently became popular in the field of data analysis because of the growing capabilities of computing technologies. With careful selection of consensus procedure, they prove to be more accurate compared to individual clustering results. Ensemble strategies benefit from combining individual runs of a component algorithm diversified in a randomized or systematic fashion. Randomized diversity is usually achieved by manipulating data (e.g., bagging and nonparametric bootstrapping), whereas systematic diversity is the result of varying parameters (e.g., parametric bootstrapping). In this paper, we employ the latter approach.

Our method, EGD, proposed in this paper is an ensemble approach that maximizes the group diffusion measure. Ensemble diversity is achieved by varying the diffusion depth parameter t. This way, the combined effect of both bias and variance error components reduction is expected. Cluster size can be bounded below by proper settings of the dependence gain parameter δ_d ranging in the interval [0, 1]. We suggest using small values of δ_d for the data sets with sparse clustering structure. On the other hand, for the data sets with dense structure and expected unclear boundaries, we recommend using higher values of δ_d. We must note that the cluster size can be bounded below by setting a hard threshold on the minimal number of points in the cluster. This provides direct intuition to setting the threshold according to the expected size of the smallest cluster. The solution with a dependence gain parameter is more flexible and acts as a form of soft threshold. The intuition for setting the dependence gain parameter is, however, less straightforward.

For evaluating the algorithm, we use both simulated and real-world data sets. In the simulated data set, EGD outperforms modularity clustering with respect to Rand and NMI measures. When small values of the width parameter σ are used, better performance is achieved for larger sets of t values. Similarly, structures of the SNAP and gene-expression data sets are best revealed by EGD with a larger set of t values. This is likely due to the non-uniform density of the underlying data, which requires consideration at different scales. On the other hand, for the NIH data set with sparse clustering structure, EGD outperforms all other methods including modularity clustering for small t values. This implies that this data set has a fine-grained structure which is better discovered by small diffusion depths. In general, this zooming mechanism is ensured by the parameter t in particular. The method is able to determine local clusters with smaller values of t, whereas higher t values allow for determining the global structure. The combination of its values allows for defining clusters more accurately.

Therefore, we conclude that EGD is suitable for solving structure discovery problems for data sets covering a wide spectrum of underlying structural and density properties thanks to flexibility in the tuning of the parameters. Diffusion depth and dependence gain parameters serve the purpose of selectively addressing data analysis at different scales, whereas the ensemble binding provides integration over the scales. The benefit of the proposed method, however, presents the questions of how one should set the parameters and of what are the theoretical interpretations, and of how one can optimally set the gain parameter depending on cluster depths, which will be a future research direction.

Despite accurate results shown in our tests, the proposed EGD algorithm can be improved further by a number of advances. In particular, we will give more detailed attention to the regularization of the algorithm's optimization criterion and the ability to efficiently handle large-sized data in the next phase of our research. Additionally, it is highly demanded to extend the method by adding the ability to determine overlapped clusters, where each instance may belong to multiple clusters simultaneously. This problem is particularly important in the field of community detection in social networks, where multiple membership is a natural attribute.

Acknowledgements

This research was supported by a grant from the HPY Research Foundation funded by Elisa, Finland. This research was also supported by the National Safety Promotion Technology Development Program (201600000002094, Smart crime prevention solution development through machine learning based on Image Big Data), funded by the Ministry of Trade, Industry and Energy(MOTIE). This work was also supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2015S1A5A2A03047963). This research was also supported by the grant (C0514059) funded by Small and Medium Business Administration (SMBA) and AURI in the Republic of Korea.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.inffus.2017.09.013

References

[1] S. Fortunato, C. Castellano, Community structure in graphs, in: Encyclopedia of Complexity and Systems Science, 2009, pp. 1141–1163.
[2] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[3] T.J. Hastie, R.J. Tibshirani, J.H. Friedman, The Elements of Statistical Learning : Data Mining, Inference, and Prediction, Springer series in statistics, Springer, New York, 2009.
[4] B. Minaei-Bidgoli, A. Topchy, W.F. Punch, Ensembles of partitions via data resampling, in: Proceedings of International Conference on Information Technology: Coding and Computing, volume 2, 2004, pp. 188–192.
[5] R. Dudoit, J. Fridly, Bagging to improve the accuracy of a clustering procedure, Bioinformatics (2003) 1090–1099.
[6] A.L.N. Fred, A.K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 835–850.
[7] J. Jia, X. Xiao, B. Liu, Similarity-based spectral clustering ensemble selection, in: Proceedings of 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012, pp. 1071–1074.
[8] A. Strehl, J. Ghosh, Cluster ensembles-a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2003) 583–617.
[9] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[10] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[11] D. Lai, H. Lu, C. Nardini, Enhanced modularity-based community detection by random walk network preprocessing, Phys. Rev. E 81 (6) (2010).
[12] A. Arenas, A. Fernandez, S. Gomez, Analysis of the structure of complex networks at different resolution levels, New J. Phys. 10 (5) (2008) 053039.
[13] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (1) (2007) 36–41.
[14] S. Fortunato, Community Detection in Graphs, Physics reports, Elsevier, 2010.
[15] H. Park, K. Lee, Dependence clustering, a method revealing community structure with group dependence, Knowl.-Based Syst. 60 (2014) 58–72.
[16] K. Lee, A. Gray, H. Kim, Dependence maps, a dimensionality reduction with dependence distance for high-dimensional data, Data Min. Knowl. Discov. 26 (3) (2013) 512–532.
[17] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[18] L. Zheng, T. Li, C. Ding, A framework for hierarchical ensemble clustering, ACM Trans. Knowl. Discov. Data 9 (2) (2014).
[19] H. Wang, H. Shan, A. Banerjee, Bayesian cluster ensembles, in: Proceedings of the 9th SIAM International Conference on Data Mining, 2009.
[20] K.D. Bollacker, J. Ghosh, Effective supra-classifiers for knowledge base construction, Pattern Recognit. Lett. 20 (11-13) (1999) 1347–1352.
[21] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, F. Warner, S. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps, in: Proceedings of the National Academy of Sciences, 2005, pp. 7426–7431.
[22] J. Ghosh, A. Acharya, Cluster ensembles, Wiley Interdiscip. Rev. 1 (4) (2011) 305–315.
[23] J. Kools, Six functions for generating artificial datasets, accessed 13 October 2016, (https://se.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets).
[24] L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, BMC Bioinform. 8 (1) (2007).
[25] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1) (1998) 359–392.
[26] J.P. Hespanha, An Efficient MATLAB Algorithm for Graph Partitioning, Technical Report, 2004.
[27] Q. Duan, Density-based clustering, accessed 20 October 2016, (https://se.mathworks.com/matlabcentral/fileexchange/53922-densityclust).
[28] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.
[29] E.A. Stone, J.F. Ayroles, Modulated modularity clustering as an exploratory tool for functional genomic inference, PLOS Genet. 5 (5) (2009) 1–13.
[30] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, in: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, in: MDS '12, ACM, New York, NY, USA, 2012.
[31] J. Leskovec, L.A. Adamic, B.A. Huberman, The dynamics of viral marketing, ACM Trans. Web (TWEB) 1 (1) (2007) 5.
[32] A. Bhattacharjee, W.G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E.J. Mark, E.S. Lander, W. Wong, B.E. Johnson, T.R. Golub, D.J. Sugarbaker, M. Meyerson, Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses, Proc. Natl. Acad. Sci. U.S.A. 98 (24) (2001) 13790–13795.
[33] E.-J. Yeoh, M.E. Ross, S.A. Shurtleff, W. Williams, D. Patel, R. Mahfouz, F.G. Behm, S.C. Raimondi, M.V. Relling, A. Patel, C. Cheng, D. Campana, D. Wilkins, X. Zhou, J. Li, H. Liu, C.-H. Pui, W.E. Evans, C. Naeve, L. Wong, J.R. Downing, Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling, Cancer Cell 1 (2) (2002) 133–143.
[34] A. Strehl, Clusterpack matlaboctave toolbox, accessed 20 October 2016, (Available at http://strehl.com/download/ClusterPackV20.zip).
[35] J.P. Hespanha, `grPartition` a MATLAB function for graph partitioning, accessed 20 October 2016, (Available at http://www.ece.ucsb.edu/~hespanha).

Supplementary Materials

Table 2: Results (p-values) of repeated measure ANOVA tests for Simulation data (the null hypothesis is $\mu_1 = \mu_2$, in which μ_i means the performance of method i, and the alternative is $\mu_1 \neq \mu_2$)

Metod1	Method2	NMI	Rand
Simulation data, $\sigma = 0.1$			
EGD ($\delta_d = 0.01$,$T = \{1\}$)	Modularity	<0.001	<0.001
	Hierarchical (k=5)	<0.001	<0.001
	Spectral (k=5)	<0.001	<0.001
	FSFDP (k=5)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001
Simulation data, $\sigma = 0.15$			
EGD ($\delta_d = 0$,$T = \{1, 2\}$)	Modularity	<0.001	<0.001
	Hierarchical (k=5)	<0.001	<0.001
	Spectral (k=5)	<0.001	<0.001
	FSFDP (k=5)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001
Simulation data, $\sigma = 0.3$			
EGD ($\delta_d = 0$,$T = \{1\}$)	Modularity	<0.001	<0.001
	Hierarchical (k=4)	<0.001	<0.001
	Spectral (k=4)	<0.001	<0.001
	FSFDP (k=5)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001

Table 3: Results (p-values) of repeated measure ANOVA tests for SNAP data (the null hypothesis is $\mu_1 = \mu_2$, in which μ_i means the performance of method i, and the alternative is $\mu_1 \neq \mu_2$)

Metod1	Method2	NMI	Rand
Amazon			
EGD ($\delta_d = 0$,T={1 to 8})	Modularity	<0.001	<0.001
	Hierarchical (k=11)	<0.001	<0.001
	Spectral (k=9)	<0.001	0.131
	FSFDP (k=11)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001
DBLP			
EGD ($\delta_d = 0.001$,T={1 to 8})	Modularity	<0.001	<0.001
	Hierarchical (k=11)	<0.001	<0.001
	Spectral (k=10)	0.300	0.043
	FSFDP (k=11)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001
YouTube			
EGD ($\delta_d = 0$,T={1 to 8})	Modularity	<0.001	<0.001
	Hierarchical (k=11)	<0.001	<0.001
	Spectral (k=10)	0.200	0.977
	FSFDP (k=11)	<0.001	<0.001
	KRF+Metis	<0.001	<0.001
	KRF+GP	<0.001	<0.001

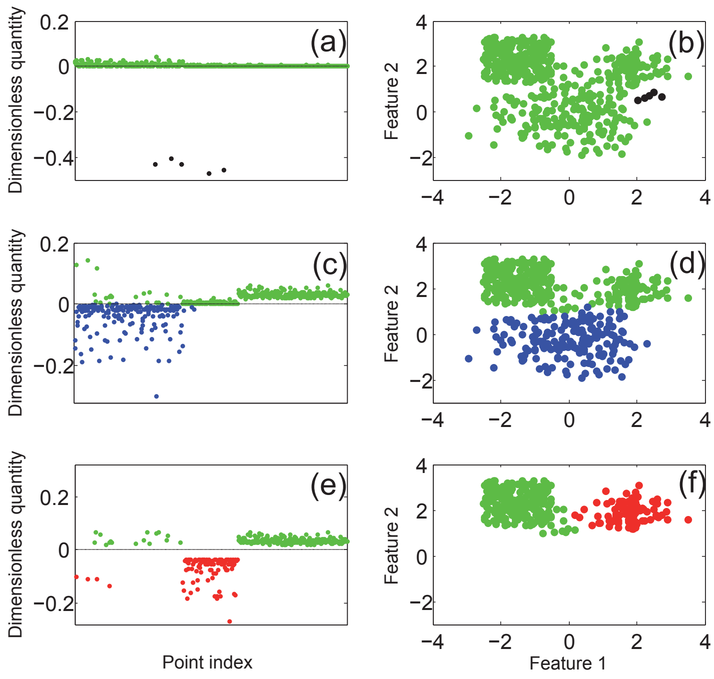


Figure 10: EGD clustering steps in illustrative example for $T = \{2, 7\}$. (a) The first eigenvector of similarity matrix at the first iteration, (b) the first division, (c) the first eigenvector of similarity matrix at the second iteration, (d) the second division, (e) the first eigenvector of similarity matrix at the third iteration, (f) the third division.

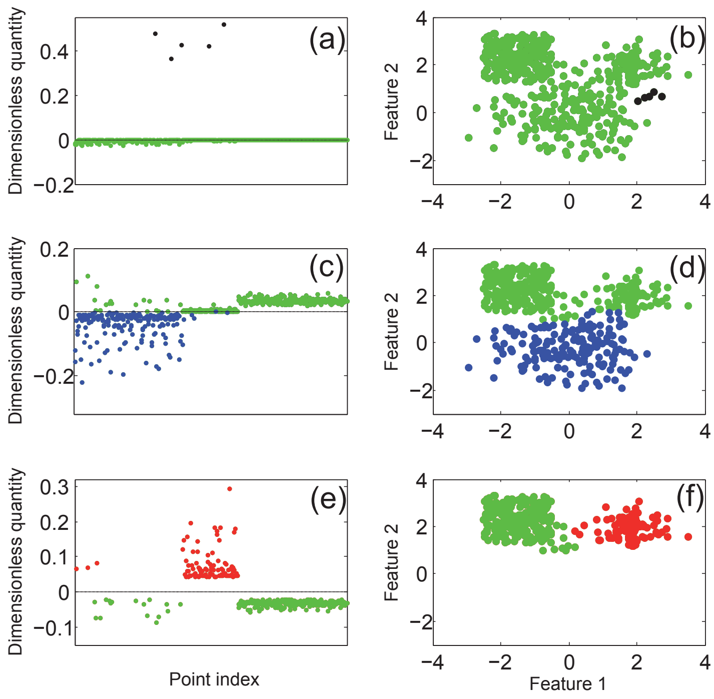


Figure 11: EGD clustering steps in illustrative example for $T = \{1, 9\}$. (a) The first eigenvector of similarity matrix at the first iteration, (b) the first division, (c) the first eigenvector of similarity matrix at the second iteration, (d) the second division, (e) the first eigenvector of similarity matrix at the third iteration, (f) the third division.

Table 4: Simulation

			$\sigma = .1$				$\sigma = .15$				$\sigma = .3$			
Method	δ_d	Parameter	CC	CS	Rand	NMI	CC	CS	Rand	NMI	CC	CS	Rand	NMI
EGD	0	$T = \{1\}$	0.8220	0.9680	0.9392	0.9073	0.8107	0.9764	0.9438	0.9115	0.8687	0.9445	**0.9295**	**0.8956**
		$T = \{1, 2\}$	0.8434	0.9598	0.9369	0.9064	0.8433	0.9695	**0.9446**	**0.9135**	0.8826	0.9305	0.9210	0.8880
		$T = \{1 \text{ to } 4\}$	0.8623	0.9509	0.9335	0.9050	0.8696	0.9578	0.9405	0.9102	0.8956	0.9083	0.9058	0.8734
		$T = \{1 \text{ to } 8\}$	0.8751	0.9432	0.9298	0.9030	0.8902	0.9434	0.9329	0.9043	0.9031	0.8824	0.8865	0.8547
	0.01	$T = \{1\}$	0.8310	0.9661	**0.9395**	**0.9079**	0.8238	0.9732	0.9438	0.9112	0.8729	0.9392	0.9261	0.8915
		$T = \{1, 2\}$	0.8511	0.9578	0.9367	0.9067	0.8520	0.9657	0.9433	0.9119	0.8863	0.9253	0.9176	0.8841
		$T = \{1 \text{ to } 4\}$	0.8677	0.9493	0.9332	0.9049	0.8753	0.9544	0.9388	0.9085	0.8984	0.9024	0.9016	0.8687
		$T = \{1 \text{ to } 8\}$	0.8807	0.9412	0.9293	0.9027	0.8936	0.9404	0.9312	0.9025	0.9067	0.8756	0.8817	0.8495
	0.2	$T = \{1\}$	0.9327	0.8520	0.8679	0.8382	0.9285	0.8393	0.8568	0.8261	0.9199	0.8179	0.8379	0.7984
		$T = \{1, 2\}$	0.9317	0.8565	0.8713	0.8421	0.9296	0.8493	0.8651	0.8352	0.9223	0.8158	0.8368	0.7989
		$T = \{1 \text{ to } 4\}$	0.9338	0.8529	0.8689	0.8403	0.9334	0.8495	0.8660	0.8374	0.9278	0.7986	0.8240	0.7887
		$T = \{1 \text{ to } 8\}$	0.9362	0.8482	0.8655	0.8377	0.9371	0.8452	0.8633	0.8358	0.9339	0.7743	0.8057	0.7745
Modularity			0.0516	0.9999	0.8131	0.7512	0.1925	0.9993	0.8404	0.7835	0.6826	0.9881	0.9279	0.8902
Hierarchical		$k = 3$	0.8861	0.7338	0.7638	0.7433	0.9236	0.7611	0.7931	0.7805	0.9487	0.7879	0.8196	0.8075
		$k = 4$	0.8131	0.9028	0.8851	0.8441	0.8484	0.9158	0.9026	0.8683	0.8690	0.9153	0.9062	0.8757
		$k = 5$	0.7156	0.9576	0.9100	0.8654	0.7356	0.9582	0.9143	0.8722	0.7408	0.9568	0.9142	0.8718
Spectral		$k = 3$	0.9148	0.7404	0.7747	0.7523	0.9175	0.7511	0.7839	0.7614	0.9271	0.7875	0.8150	0.7905
		$k = 4$	0.8361	0.8689	0.8624	0.8236	0.8406	0.8811	0.8731	0.8348	0.8530	0.9124	0.9007	0.8632
		$k = 5$	0.7276	0.9309	0.8908	0.8421	0.7308	0.9407	0.8993	0.8515	0.7271	0.9550	0.9101	0.8624

Continued on next page

Table 4: Simulation – continued from previous page

			$\sigma = .1$				$\sigma = .15$				$\sigma = .3$			
Method	δ_d	Parameter	CC	CS	Rand	NMI	CC	CS	Rand	NMI	CC	CS	Rand	NMI
FSFDP		$k = 3$	0.8750	0.6123	0.6640	0.6482	0.8935	0.6454	0.6942	0.6835	0.9281	0.6488	0.7038	0.6923
		$k = 4$	0.8315	0.6618	0.6952	0.6707	0.8493	0.6954	0.7257	0.7078	0.8984	0.6894	0.7306	0.7138
		$k = 5$	0.7958	0.6935	0.7137	0.6828	0.8044	0.7249	0.7405	0.7167	0.8703	0.7130	0.7440	0.7235
Metis		$k = 3$	0.8467	0.8491	0.8486	0.7748	0.8476	0.8511	0.8504	0.7777	0.8444	0.8506	0.8494	0.7739
		$k = 4$	0.8148	0.9532	0.9260	0.8752	0.8085	0.9554	0.9264	0.8763	0.7946	0.9528	0.9216	0.8688
		$k = 5$	0.6647	0.9754	0.9142	0.8643	0.6680	0.9782	0.9171	0.8685	0.6566	0.9744	0.9118	0.8584
KRF+Metis			0.6781	0.9738	0.9155	0.8675	0.6774	0.9767	0.9177	0.8697	0.6675	0.9733	0.9131	0.8612
GP		$k = 3$	0.9115	0.7460	0.7786	0.7532	0.9125	0.7586	0.7889	0.7627	0.9139	0.7808	0.8070	0.7778
		$k = 4$	0.8269	0.8726	0.8636	0.8217	0.8268	0.8855	0.8739	0.8319	0.8296	0.8996	0.8858	0.8428
		$k = 5$	0.7174	0.9362	0.8931	0.8422	0.7153	0.9450	0.8997	0.8491	0.7104	0.9487	0.9017	0.8495
KRF+GP			0.6948	0.9446	0.8954	0.8402	0.7013	0.9536	0.9039	0.8520	0.6952	0.9581	0.9063	0.8537

Table 5: NIH

Method	δ_d	Parameter	CC	CS	Rand	NMI
EGD	0	$T = \{1\}$	0.3422	0.9306	**0.9201**	**0.4946**
		$T = \{1, 2\}$	0.3253	0.9305	0.9192	0.4868
		$T = \{1$ to $4\}$	0.2907	0.9311	0.9156	0.4743
		$T = \{1$ to $8\}$	0.2915	0.9312	0.9155	0.4726
	0.0001	$T = \{1\}$	0.3313	0.9306	0.9195	0.4881
		$T = \{1, 2\}$	0.3253	0.9305	0.9192	0.4868
		$T = \{1$ to $4\}$	0.2907	0.9311	0.9156	0.4743
		$T = \{1$ to $8\}$	0.2915	0.9312	0.9155	0.4726
Modularity			0.3048	0.9308	0.9173	0.4843
Hierarchical		$k = 43$	0.0753	0.9303	0.2287	0.1690
		$k = 44$	0.0753	0.9303	0.2287	0.1699
		$k = 45$	0.0754	0.9307	0.2305	0.1717
Spectral		$k = 43$	0.1322	0.9299	0.8767	0.3571
		$k = 44$	0.1424	0.9303	0.8810	0.3705
		$k = 45$	0.1309	0.9295	0.8797	0.3523
FSFDP		$k = 43$	0.0777	0.9612	0.1559	0.1686
		$k = 44$	0.0771	0.9547	0.1565	0.1519
		$k = 45$	0.0773	0.9504	0.1745	0.1612
Metis		$k = 43$	0.2436	0.9296	0.9142	0.4093
		$k = 44$	0.2703	0.9300	0.9157	0.4296
		$k = 45$	0.2533	0.9296	0.9152	0.4189
KRF+Metis			0.2505	0.9295	0.9150	0.4134
GP		$k = 43$	0.1568	0.9297	0.8938	0.3676
		$k = 44$	0.1430	0.9294	0.8893	0.3613
		$k = 45$	0.1694	0.9299	0.8975	0.3788
KRF+GP			0.2150	0.9287	0.9135	0.3800

Table 6: SNAP

			Amazon				DBLP				YouTube			
Method	δ_d	Parameter	CC	CS	Rand	NMI	CC	CS	Rand	NMI	CC	CS	Rand	NMI
EGD	0	$T = \{1\}$	0.7056	1	0.9331	0.9290	0.7269	0.9998	0.9456	0.9357	0.4743	0.9978	0.8470	0.8311
		$T = \{1, 2\}$	0.7479	1	0.9410	0.9394	0.7959	0.9999	0.9537	0.9513	0.6607	0.9960	0.8851	0.8841
		$T = \{1 \text{ to } 4\}$	0.8042	1	0.9516	0.9535	0.8518	0.9999	0.9609	0.9635	0.8117	0.9942	0.9261	0.9235
		$T = \{1 \text{ to } 8\}$	0.8661	1	**0.9643**	**0.9673**	0.8935	1	0.9674	0.9732	0.9031	0.9886	0.9554	**0.9473**
	0.001	$T = \{1\}$	0.7135	1	0.9346	0.9315	0.7447	0.9998	0.9481	0.9408	0.4949	0.9963	0.8511	0.8346
		$T = \{1, 2\}$	0.7603	0.9999	0.9432	0.9425	0.8093	0.9998	0.9554	0.9545	0.6706	0.9955	0.8878	0.8845
		$T = \{1 \text{ to } 4\}$	0.8122	0.9999	0.9528	0.9550	0.8565	0.9999	0.9617	0.9648	0.8150	0.9935	0.9267	0.9204
		$T = \{1 \text{ to } 8\}$	0.8661	0.9999	**0.9643**	0.9667	0.8957	1	0.9679	**0.9740**	0.9062	0.9872	**0.9557**	0.9446
	0.01	$T = \{1\}$	0.7884	0.9853	0.9366	0.9117	0.8441	0.9946	0.9569	0.9497	0.6091	0.9787	0.8648	0.8229
		$T = \{1, 2\}$	0.8084	0.9877	0.9419	0.9200	0.8663	0.9949	0.9601	0.9547	0.7580	0.9770	0.9004	0.8638
		$T = \{1 \text{ to } 4\}$	0.8543	0.9865	0.9507	0.9274	0.8984	0.9949	0.9650	0.9608	0.8538	0.9797	0.9308	0.8925
		$T = \{1 \text{ to } 8\}$	.8846	0.9899	0.9599	0.9424	0.9122	0.9958	**0.9682**	0.9658	0.9252	0.9729	0.9508	0.9083
Modularity			0.7916	1	0.9504	0.9501	0.8455	0.9999	0.9610	0.9624	0.7238	0.9937	0.9017	0.9019
Hierarchical		$k = 9$	0.7843	0.3300	0.4209	0.3085	0.7966	0.6917	0.7157	0.6687	0.7609	0.2146	0.3662	0.1839
		$k = 10$	0.7600	0.3553	0.4376	0.3239	0.7728	0.7227	0.7396	0.6847	0.744	0.2396	0.3822	0.2006
		$k = 11$	0.7306	0.3881	0.4598	0.3444	0.7457	0.7537	0.7627	0.6996	0.7269	0.2666	0.4001	0.2182
Spectral		$k = 9$	0.7907	0.9282	0.8951	0.8473	0.8234	0.9041	0.8800	0.8385	0.7029	0.9336	0.8621	0.8063
		$k = 10$	0.7243	0.9449	0.8960	0.8458	0.8088	0.9279	0.8959	0.8606	0.7102	0.9767	0.8977	0.8625
		$k = 11$	0.6320	0.9479	0.8837	0.8271	0.7175	0.9333	0.8918	0.8449	0.6043	0.9775	0.8718	0.833

Continued on next page

Table 6: SNAP – continued from previous page

			Amazon				DBLP				YouTube			
Method	δ_d	Parameter	CC	CS	Rand	NMI	CC	CS	Rand	NMI	CC	CS	Rand	NMI
FSFDP		$k = 9$	0.6960	0.5464	0.5781	0.4942	0.7418	0.6198	0.6398	0.5845	0.6863	0.4495	0.5133	0.3788
		$k = 10$	0.6531	0.5977	0.6087	0.5182	0.7203	0.6572	0.6686	0.6044	0.6616	0.4728	0.5268	0.3866
		$k = 11$	0.6454	0.6201	0.6280	0.5341	0.7020	0.6751	0.6818	0.6115	0.6398	0.5103	0.5480	0.4086
KRF+Metis		$k = 9$	0.5117	0.9716	0.8803	0.7931	0.6836	0.9707	0.9174	0.8422	0.2436	0.9296	0.9142	0.4093
		$k = 10$	0.4744	0.9779	0.8793	0.7953	0.6454	0.9781	0.9192	0.8482	0.2703	0.9300	0.9157	0.4296
		$k = 11$	0.4396	0.9822	0.8773	0.7948	0.6221	0.9852	0.9225	0.8619	0.2533	0.9296	0.9152	0.4189
			0.4442	0.9808	0.8769	0.7883	0.6261	0.9835	0.9213	0.8543	0.2505	0.9295	0.9150	0.4134
KRF+GP		$k = 9$	0.4818	0.8880	0.8095	0.6497	0.5280	0.8435	0.7908	0.6252	0.1568	0.9297	0.8938	0.3676
		$k = 10$	0.4500	0.9086	0.8203	0.6619	0.4882	0.8547	0.7964	0.6169	0.1430	0.9294	0.8893	0.3613
		$k = 11$	0.4137	0.9229	0.8263	0.6667	0.4800	0.8820	0.8182	0.6474	0.1694	0.9299	0.8975	0.3788
			0.3492	0.9605	0.8464	0.6652	0.3922	0.9527	0.8689	0.6534	0.2150	0.9287	0.9135	0.3800

Table 7: Amazon co-purchasing relationships

Method	δ_d	Parameter	CC	CS	Rand	NMI
EGD	0	$T = \{1\}$	0.0772	0.9184	0.8317	**0.2971**
		$T = \{1, 2\}$	0.0813	0.9192	0.7697	0.2563
		$T = \{1 \text{ to } 4\}$	0.0810	0.9185	0.8190	0.2746
		$T = \{1 \text{ to } 8\}$	0.0808	0.9188	0.7680	0.2326
	0.001	$T = \{1\}$	0.0800	0.9185	0.8179	0.2564
		$T = \{1, 2\}$	0.0816	0.9192	0.7616	0.2253
		$T = \{1 \text{ to } 4\}$	0.0804	0.9184	0.8116	0.2381
		$T = \{1 \text{ to } 8\}$	0.0804	0.9187	0.7641	0.2065
Modularity			0.0778	0.9182	**0.8441**	0.2872

Table 8: Lung cancer

Method	δ_d	Parameter	CC	CS	Rand	NMI
EGD	0.09	$T = \{1\}$	0.2928	0.7630	0.5164	0.1069
		$T = \{1, 2\}$	0.2172	0.7902	0.4896	0.1259
		$T = \{1 \text{ to } 4\}$	0.2643	0.7477	0.4941	0.0345
		$T = \{1 \text{ to } 8\}$	0.3346	0.7997	**0.5557**	**0.3609**
Modularity			0	1	0.4754	**0.4192**
Hierarchical		$k = 3$	0.9963	0.0386	0.5410	0.0974
		$k = 4$	0.9947	0.0578	0.5493	0.1206
		$k = 5$	0.9931	0.0771	**0.5576**	0.1408
Spectral		$k = 3$	0.4330	0.6147	0.5194	0.0188
		$k = 4$	0.4162	0.6056	0.5063	0.0153
		$k = 5$	0.5231	0.4420	0.4845	0.0508
FSFDP		$k = 3$	0.9864	0.0063	0.5204	0.0103
		$k = 4$	0.9864	0.0063	0.5204	0.0103
		$k = 5$	0.9864	0.0063	0.5204	0.0103
Metis		$k = 3$	0.3281	0.6680	0.4897	0.0061
		$k = 4$	0.2435	0.7508	0.4847	0.0083
		$k = 5$	0.1952	0.8032	0.4843	0.0160
KRF+Metis			1	0	0.5246	0
GP		$k = 3$	0.3292	0.6675	0.4901	0.0137
		$k = 4$	0.2512	0.7543	0.4904	0.0246
		$k = 5$	0.1986	0.7994	0.4843	0.0493
KRF+GP			1	0	0.5246	0

Table 9: St. Jude leukemia

Method	δ_d	Parameter	CC	CS	Rand	NMI
EGD	0.06	$T = \{1\}$	0.9469	0.9319	0.9351	0.8262
		$T = \{1, 2\}$	0.9469	0.9623	0.9589	**0.8800**
		$T = \{1 \text{ to } 4\}$	0.8882	0.9648	0.9482	0.8539
		$T = \{1 \text{ to } 8\}$	0.7514	0.9827	0.9325	0.8172
Modularity			0.9686	0.8411	0.8688	0.7640
Hierarchical		$k = 3$	0.9512	0.0475	0.2436	0.0845
		$k = 4$	0.9485	0.0568	0.2503	0.0974
		$k = 5$	0.9482	0.0568	0.2503	0.0953
Spectral		$k = 3$	0.9418	0.9646	**0.9596**	0.8663
		$k = 4$	0.7265	0.9640	0.9125	0.7675
		$k = 5$	0.8488	0.9879	0.9578	0.8640
FSFDP		$k = 3$	0.9449	0.7988	0.8305	0.7428
		$k = 4$	0.9233	0.8833	0.8920	0.8253
		$k = 5$	0.9175	0.8761	0.8851	0.8038
Metis		$k = 3$	0.9497	0.3772	0.5015	0.4488
		$k = 4$	0.7265	0.7412	0.7380	0.4956
		$k = 5$	0.6348	0.6824	0.6721	0.4129
KRF+Metis			1	0	0.2170	0
GP		$k = 3$	0.6333	0.9242	0.8611	0.6845
		$k = 4$	0.5557	0.9454	0.8608	0.6704
		$k = 5$	0.4956	0.9593	0.8587	0.6764
KRF+GP			1	0	0.2170	0

PIV

PROBABILISTIC GROUP DEPENDENCE APPROACH FOR DISCOVERING OVERLAPPING CLUSTERS

by

Elena Ivannikova, Anna V Kononova, and Timo Hämäläinen 2016

Proc. of the 26th International Workshop on Machine Learning for Signal Processing, pp. 1-6

PROBABILISTIC GROUP DEPENDENCE APPROACH FOR DISCOVERING OVERLAPPING CLUSTERS

Elena Ivannikova[a] *Anna V. Kononova*[b] *Timö Hämäläinen*[a]

[a]Department of Mathematical Information Technology
University of Jyväskylä
POBox 35 (Agora), 40014 Jyväskylä, Finland
elena.v.ivannikova@student.jyu.fi
timo.t.hamalainen@jyu.fi

[b]MACS
Heriot-Watt University
Edinburgh, EH14 4AS, UK
anna.kononova@gmail.com

ABSTRACT

This article proposes Soft Dependence Clustering (SDC) algorithm which belongs to the class of spectral clustering methods. On each iteration, SDC performs a hierarchical clustering producing a binary split which greedily maximizes the group dependence score. One of the advantages of SDC is the fact that division of a group into two clusters is done based on the adjustable threshold which has a clear probabilistic interpretation. Due to this property, the algorithm naturally allows fuzzy group separations which makes it also suitable for cluster overlaps analysis. SDC can be used for graph segmentation applications as well as clustering data that has notion of distance. The proposed algorithm is compared with a few selected clustering methods using simulated and real-world data sets. The results clearly demonstrate that given reasonable settings, SDC outperforms other methods in the comparison.

Index Terms— Soft clustering, spectral analysis, random walk, Markov chain

1. INTRODUCTION

Over the recent years, there has been a significant increase in the interest towards detecting overlapping clusters in data. Despite the majority of clustering algorithms being designed for hard clustering only, the ability to identify cluster overlaps is crucial for many real-world data sets. Detecting such cluster overlaps in data provides a better understanding of both the structure and nature of the underlying application areas since in many of them data set items can naturally belong to multiple clusters. An example of such situation is the interplay between multiple genes that regulate biological processes – for many of them, ultimate discovery of the overlapping groups of genes provides a more realistic model of cellular activity, compared to the mutually exclusive clustering [1]. Similarly, many real-world text data sets have complex structure where items belong to multiple groups. For example, in Reuters data set [2], the newswire stories are grouped based on the category codes and assigned to several highly unbalanced classes. Another example are network communities where a node can belong simultaneously to several communities leading to overlapping community structure.

To tackle problems with overlapping clusters, a number of clustering schemes have been developed. One frequently used approach is running a well-known clustering method and adjusting the result to produce overlapping clusters, e.g. through introducing a threshold parameter. However, such an approach has limitations. First, constraints for the objective function being optimized do not match the expected clustering scheme. Second, the setting of the *global* threshold value remains an open question [3]. Therefore, some common models have been generalized to allow overlapping clusters [3], [4].

In this paper, we propose Soft Dependence Clustering (SDC) algorithm which considers geometric structure of data and is based on maximizing a measure called *group dependence* [5]. This measure provides flexibility in adjusting the level of detail and connectivity scale in network analysis. Park and Lee [6] demonstrated efficacy of similar method for clustering into distinct groups. The proposed SDC algorithm is therefore a generalization of the dependence clustering method which supports soft clustering. Performance of the method proposed here is compared with Spectral Fuzzy C-Means (SFCM) [7] and Latent Dirihlet Allocation (LDA) [8] where the first is a soft version of k-Means applied after spectral decomposition of a random-walk transition matrix and the second is a common choice for processing text data sets. These methods are known to be good at discovering clusters in data where overlapping clusters assumption is natural.

The rest of the paper is organized as follows. Section 2 provides descriptions of the main concepts and methods used in the paper. Sections 2.1 - 2.4 explain the proposed SDC clustering algorithm. Meanwhile, SFCM and LDA methods used for comparison with SDC are provided in Sections 2.5

and 2.6, respectively. Section 3 is devoted to the experimental results. It describes evaluation metrics, data sets, parameter estimation and results of the performance tests. Finally, Section 4 completes the paper with conclusions and discussions.

2. METODS

2.1. Preliminary concepts

Given a set of data points $\Omega = \{x_i | i = 1, ..., N; x_i \in \mathbb{R}^n\}$, we assume that points from the set Ω form a graph defined by a similariy matrix S of size $N \times N$ with entries from $\mathbb{R}$ representing pairwise similarities. Next, we define the Markov chain on this graph by transforming S to the transition matrix P and the corresponding t-step transition matrix $\mathrm{P}^t : P^t_{i,j} = Pr(X_t = j|X_0 = i)$, where a probability variable X_0 represents the initial state and X_t is a random walk representing a node at the t-th transition. The transformation is effectively done by scaling rows of S so that elements in each row sum up to one. Further, we assume that the whole chain is ergodic and that all transitions follow the Markovian property. Statistical dependence $D_{i,j,t} = Dep(X_0 = i, X_t = j)$ [5] captures how the node in the initial state and the node at t-th transition are inter-dependent and is defined by the following equation:

$$D_{i,j,t} = \frac{Pr(X_0 = i, X_t = j)}{Pr(X_0 = i)Pr(X_t = j)}. \quad (1)$$

Let us denote a group assignment vector by $\mathbf{s} = [s_1, ..., s_N]$, where decision variable $s_i = 1$ if data point i belongs to group 1 and $s_i = -1$ if it belongs to group 2. Note that in such notation $1/2(s_i s_j + 1)$ is 1 if i and j are in the same group and is 0, otherwise. Thus, given s and t, the group dependence for a particular choice of s is defined by the following equation:

$$D_t(\mathbf{s}) = 1/2 \sum_{x_i, x_j \in \Omega} \big(D_{i,j,t} - d_0\big)(s_i s_j + 1), \quad (2)$$

where $D_{i,j,t}$ is defined by (1) and $d_0 = 1 + \epsilon_d$ is the baseline dependence level which is usually set to 1. A constant d_0 effectively normalizes the statistical dependence for each pair of points equaling zero when points are considered to be independent. More details about parameter settings and optimization procedure can be found in [6].

2.2. Dependence clustering for two groups

For a simple case of bisecting a graph, an optimal clustering solution can be achieved through maximizing the group dependence measure $D_t(\mathbf{s})$ by varying the group assignment s of all N points. The actual optimization is carried out in the domain of real numbers $\mathbb{R}$ by relaxing the original formulation (2) so that elements of s become real. Moreover, we constrain the L_2 norm of s to be equal to one: $||\mathbf{s}||_2 = 1$ and assume for simplicity that $\epsilon_d = 0$. Then, a good partition is obtained through solving the following maximization problem [6]:

$$\begin{aligned} \arg\max_{||\mathbf{s}||=1} D_t(\mathbf{s}) &= \arg\max_{||\mathbf{s}||=1} 1/2 \sum_{i,j} (D_{i,j,t} - 1)(s_i s_j + 1) \\ &= \arg\max_{||\mathbf{s}||=1} \mathbf{s}^T (\mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T)\mathbf{s} \quad (3) \end{aligned}$$

where $\mathrm{B}^{(t)} : B^{(t)}_{j,j} = Pr(X_t = j) = [x_0^T \mathrm{P}^t]_j$, and x_0 is the initial probability vector representing prior information related to the initial states. Finally, let us define

$$\mathrm{G} = \mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T. \quad (4)$$

Then, division of the data points is made based on the signs of the eigenvector corresponding to the largest positive eigenvalue of G. The nonexistence of positive eigenvalues means no possible benefits from any further divisions.

2.3. Dependence clustering for multiple groups

To obtain divisions to multiple groups, we apply a standard subsequent division approach [9]. At every step we consider binary divisions of every group already found during the previous iterations, following the algorithm described in Section 2.2. Among possible divisions we proceed with the one that results in the maximal increase of group dependence for the whole data set, effectively performing a greedy search.

For the purpose of regularizing the solution, we introduce a dependence gain parameter $\delta_d \in [0, 1]$, which prevents the algorithm from defining too small or too unclear clusters. The minimal dependence gain required to split a cluster into two sub-clusters is calculated as $\Delta_d = \delta_d \sum_{g_{i,j} > 0} G$, where G is defined by (4). Let us denote a set of points that belong to a split candidate cluster by $\Omega_C \subseteq \Omega$. We denote the within-cluster group configurations before and after the division of Ω_C into two sub-clusters by $\mathbf{s}_C$ (i.e. s in (2)) and $\mathbf{s}'_C$ (i.e. s in (3)), respectively. The division into the sub-clusters proceeds if $D_t(\mathrm{sgn}(\mathbf{s}'_C)) - D_t(\mathbf{s}_C) > N\Delta_d$, where $D_t(\mathbf{s})$ is defined by (2) and N is the size of Ω_C. Note that all elements from $\mathbf{s}_C$ are equal to one. We denote $D_t(\Omega_C) = D_t(\mathbf{s}_C)$ when all elements from $\mathbf{s}_C$ have the same sign.

2.4. SDC

Dependence clustering has alreardy demonstrated accurate results compared to a number of methods [6]. However, it provides only a hard clustering scheme and is not designed for handling overlaps. SDC algorithm proposed here extends the earlier scheme from [6] by introducing a *soft probability interval*.

Note that the binary split of data points on every iteration of the dependence clustering algorithm is made based on the signs of the eigenvector corresponding to the largest positive eigenvalue of G defined in (4). To extend the basic algorithm to perform soft clustering, we therefore need to re-define the split decision to be soft itself. In other words, we need to introduce the uncertainty region [0, δ_u] which filters only instances belonging to positive split if values of their corresponding decision eigenvector belong to this interval.

On the other hand, this soft decision interval should be adjusted on every iteration depending on a particular eigenvector configuration. A solution to this would be to interpret eigenvector values as generated from two probability distributions: one with a negative mean and the other with a positive mean. Having estimated such probabilities, one could then introduce a *soft probability interval* $\delta_P = [p_l, p_h]$, such that $p_l + p_h = 1$. This interval defines the lower and the upper bounds of the posterior probability that a data point should have in order to be assigned to both clusters.

In our implementation, posterior probabilities are computed from likelihood probability distributions assuming equal priors for both clusters, i.e. by normalizing the likelihoods sum to one. The two probability distributions are estimated by Gaussian kernel density estimation with standard deviation or bandwidth σ. Figure 1 displays pseudo code summarizing the above steps.

2.5. SFCM

The basic FCM [7] algorithm works as follows. We denote a number of clusters by N_c, centers of the corresponding clusters by $\mathbf{c} = (c_1, ..., c_{N_c})$, the degree of membership of the point x_i in the cluster c_j by $\mu_{ij} : \sum_{j=1}^{N_c} \mu_{ij} = 1$ and the degree of fuzzy overlaps by $m > 0$. Given the parameters N_c, m and initialization values for μ_{ij} FCM minimizes the objective function: $J_m = \sum_{i=1}^{N} \sum_{j=1}^{N_c} \mu_{ij}^m \parallel x_i - c_j \parallel^2$, where $c_j = \sum_{i=1}^{N} \mu_{ij}^m x_i / \sum_{i=1}^{N} \mu_{ij}^m$, $\mu_{ij} = 1/\sum_{k=1}^{N_c}(\parallel x_i - c_j \parallel / \parallel x_i - c_k \parallel)^{2/(m-1)}$ are recalculated at every iteration. Performing clustering in the original $\mathbb{R}^n$ space can be non-optimal especially if n is large. Therefore, we do dimensionality reduction, first. This step is done by spectral decomposition of the t-step transition matrix P^t obtained from the similarity matrix S defined in Section 2.1. Only top N_v eigenvectors with the largest eigenvalues are retained, excluding the first one that equals to the unit vector [10]. Then FCM is applied to the selected N_v eigenvectors.

Thus, cluster attribution decision goes as follows. First, each data point is assigned to a cluster with the largest membership value. Moreover, point x_i belongs to both clusters c_j and c_k if its membership values satisfy the following criteria:

$$\begin{cases} |\mu_{ij} - \mu_{ik}| < \Delta_-, \\ \mu_{ij} + \mu_{ik} > \frac{2}{N} + \Delta_+, \end{cases}$$

where Δ_- and Δ_+ are free parameters that define maximal difference and minimal sum of the two memberships in order to declare a data instance as belonging to both clusters. In our experimental setup, we use MathWorks fuzzy logic toolbox [11] for FCM computation.

```
Require: Set of data points $\Omega$, similarity matrix S, parameters $t$, $\delta_d$, $\epsilon_d$, $\delta_P$, $\sigma$.
  Initialize a set $\Psi = \{\Omega\}$, $\Delta_d$ as described in Section 2.3;
  Compute transition matrix P from S (see Section 2.1);
  while true do
    Set $\Omega'_C = \emptyset$, $\Omega' = \emptyset$;
    for all $\Omega_C$ in $\Psi$ do
      Find a sub-division $\mathbf{s}'_C$ of $\Omega_C$ by solving (3);
      Fit two probability distributions $P(x_i|-)$, $P(x_i|+)$ using Gaussian kernel density estimation with bandwidth $\sigma$ applied to negative and positive values of the eigenvector $\mathbf{s}'_C$ correspondingly.
      Estimate posterior probabilities for $+$ and $-$ classes given $x_i$ as $P(-|x_i) = P(x_i|-)/(P(x_i|-) + P(x_i|+))$, $P(+|x_i) = 1 - P(-|x_i)$;
      Define $\Omega^1_C = \{x_i|\, \mathrm{sgn}(\mathbf{s}'_C(i)) > 0\} \cup \{x_i|P(-|x_i) \in \delta_P, P(+|x_i) \in \delta_P\}$ and $\Omega^2_C = \{x_i|\, \mathrm{sgn}(\mathbf{s}'_C(i)) \le 0\} \cup \{x_i|P(-|x_i) \in \delta_P, P(+|x_i) \in \delta_P\}$;
      if $D_t(\Omega^1_C) + D_t(\Omega^2_C) - D_t(\Omega_C) > N\Delta_d$ then
        $\Omega'_C = \Omega_C$, $\Omega' = \{\Omega^1_C, \Omega^2_C\}$;
      end if
    end for
    if $\Omega'_C \neq \emptyset$ then
      Apply the sub-division of $\Omega'_C$ by replacing $\Omega'_C$ in $\Psi$ by the two elements of the set $\Omega'$;
    else
      Finish the loop.
    end if
  end while
```

Fig. 1. Pseudo code for SDC algorithm

2.6. LDA

In LDA formulation [8] we assume that the number of topics K is given beforehand and fixed, the number of words in the vocabulary is V, the number of documents is N and the number of words in d-th document is N_d. Let us denote a set of documents by $\mathbf{w} = (w_1, ..., w_N)$, where $w_d = (w_{d,1}, ..., w_{d,N_d})$ defines a sequence of words in d-th document. We denote topics assigned to words by $\mathbf{z} = (z_1, ..., z_N)$, where each $z_d = (z_{d,1}, ..., z_{d,N_d})$ corresponds to w_d. Suppose, $\boldsymbol{\beta} = \{\beta_i | i = 1, ..., V\}$ is a collection of parameters of Dirichlet distribution [8] corresponding to prior weights of words in a topic and $\boldsymbol{\alpha} = \{\alpha_i | i = 1, ..., K\}$ is a collection of parameters of Dirichlet distribution corresponding to prior weights of topics in a document. Given parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the joint probability of $\mathbf{w}$, $\mathbf{z}$, $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ is

$$P(\mathbf{w}, \mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\varphi} | \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{i=1}^{K} P(\varphi_i | \boldsymbol{\beta}) \prod_{d=1}^{N} P(\theta_d | \boldsymbol{\alpha}) \times \\ \times \Big(\prod_{n=1}^{N_d} P(z_{d,n} | \theta_d) P(w_{d,n} | \boldsymbol{\varphi}, z_{d,n}) \Big),$$

where $\theta_d \sim \mathrm{Dir}_K(\boldsymbol{\alpha})$ is a multinomial distribution of topics in document d, $\varphi_i \sim \mathrm{Dir}_V(\boldsymbol{\beta})$ is a multinomial distribution of words in topic i, $z_{d,n}$ is identity of topic of word n in document d sampled from θ_d and $w_{d,n}$ is identity of word n in document d sampled from $\varphi_{z_{d,n}}$. Then, LDA problem consists in inferring the latent variables $\mathbf{z}$, θ and φ that maximize posterior distribution $P(\mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\varphi} | \mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ [8].

To estimate performance of the LDA, we use a Matlab Topic Modeling toolbox [12]. We adapted this implementation for discovering overlapping clusters of documents. First, we extract topics with LDA model by estimating $\mathbf{z}$, θ and φ. Then, to every document we assign a label corresponding to the topic which received maximal probability p_{max} over all topics. In addition, we assign to every document all other topics with probabilities p_i such that $p_i \geq p_{max} - \delta$. Here, the probability margin parameter δ denotes the maximal probability difference between the winner topic probability p_{max} and a probability of a candidate topic p_i where $i = 1, ..., K$.

3. EXPERIMENTAL RESULTS

3.1. Evaluation metrics

Given the true clustering labels, we use the following evaluation approaches to measure performance of the algorithms:
Normalized Mutual Information (NMI) is an information theory based metric used to compare true labels and predicted results - employed here as described in [13].
Average $\mathbf{F_1}$-score computed [14] as $F_1 = 1/2(F_{1t} + F_{1d})$, where F_{1t} is F_1-score of best-matching true label to each detected label and F_{1d} is F_1-score of best-matching detected label to each true label. In addition, average F_{1s} and F_{1m} scores are introduced that are calculated similarly to F_1-score and show how well an algorithm clusters the true single-labeled and true multi-labeled instances, respectively.

3.2. Demonstration example

For a demo example, we construct a similarity matrix by randomly generating 600 points in the two dimensional space where every point belongs to one of three clusters. Each of these clusters is formed by a Gaussian distribution with covariance matrix $0.5 \times \mathbf{I}$ and consists of 200 points. Two clusters are shifted from the center of first cluster by approximately 1.7 units in the opposite directions along the axis x. The obtained data set is shown in Figure 2(a), where each cluster is displayed in its own color. It is clearly visible that the border is fuzzy and the points in its vicinity are most likely to have an overlapping nature.

The proposed SDC algorithm is applied to the demo example data set. Figure 2(b) shows clustering results with SDC for the parameters $\sigma = 0.01$, $\epsilon_d = 0.1$, $t = 1$, $\delta_d = 0.1$ and the probability interval $\delta_P = [0.4, 0.6]$. The clusters corresponding to original data clusters are displayed in similar colors. Discovered overlapping instances are marked by black circles. The method successfully determines all three clusters and marks points near the identified border as overlaps. The

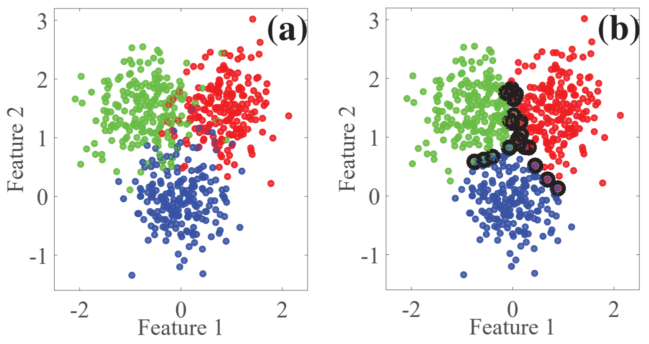


Fig. 2. (a) The original data set and true cluster division from demo example, (b) SDC clustering results for the demo example. Black circles denote overlapping instances.

results of SFCM for this data set are similar to the results of SDC clustering algorithm. This is likely due to the data fitting well the underlying assumptions of the methods. Therefore, we omit the results of SFCM here.

3.3. Reuters data set

In our tests we refer to the tag 'Topics' of the Reuters data set [2]. Four topics were selected: *grain*, *crude*, *interest* and *trade*. Initially, only the first 2000 articles were taken where at least one of the selected topics was present. Similarity matrix for the articles was defined as a cosine similarity among L_2-normalized term frequency (TF) document vectors.

First, we performed feature selection by extracting a subset of articles where only one of the selected topics was present. Next, we generated TF features following the procedure of removing punctuation, tokenization, stemming (Porter stemmer [15]) and ignoring all terms that appear in more than 40% of documents. Among the obtained TF features we selected the 500 most discriminative features according to the χ^2 test [16], which effectively resulted in a reduced vocabulary of 500 terms. We repeated the same procedure of generating TF matrix for each of the 2000 documents considering only terms from the reduced vocabulary. Finally, we obtained 1876 articles with at least one non-zero feature and built a cosine similarity matrix S used as an input for the tested algorithms.

In our experiments, we were interested in discovering parameter settings for which the tested algorithms produced the best clustering results in terms of selected average F_1-score and NMI cluster quality measures. These best parameter values were selected based on the results of a grid search (see Table 1 for the parameter grid settings). The performance results are displayed in Table 2 with the highest metrics' values marked in bold. Presence of two lines for a method in Table 2 signifies that NMI and average F_1-score have attained their maximum values for different sets of parameters. As one can see from this table, SDC outperforms other methods under all their parameter settings from Table 1. We would like to draw

Table 1. Parameter grid settings for the Reuters and Gene data sets. Intervals are denoted with square brackets.

		Reuters	Gene
SDC	δ_P	[0.2, 0.8]	[0.2, 0.8]
	δ_d	[0.01, 0.2], step 0.01	[0.01, 0.1], step 0.005
	ϵ_d	[0.0, 0.2], step 0.01	[10^{-7},10^{-5}], step 10^{-7}
	σ	0.01	0.01
	t	[2,3], step 1	[2,4], step 1
SFCM	Δ_-	[0.01, 0.2], step 0.01	[0.01, 0.1], step 0.005
	Δ_+	[-0.05, 0.2], step 0.01	[0, 0.1], step 0.01
	N_c	[4,8], step 1	[12, 27], step 1
	m	3	[3,5], step 1
	t	2	[2,4], step 1
LDA	α	[2.5, 25], step 2.5	[1,10], step 0.01
	β	[0.005,0.01],step 0.005	[0.01,0.2], step 0.01
	δ	[0.001,0.3], step 0.001	[0.05,0.3], step 0.05
	K	[3,5], step 1	[10,27], step 1

Table 2. Reuters results

Parameters		NMI	F_1	F_{1s}	F_{1m}
SDC	*	0.5994	**0.8680**	**0.8702**	**0.7631**
	**	**0.6116**	0.8613	0.8638	0.7080
SFCM	***	0.5592	0.8250	0.8291	0.5147
LDA	****	0.5413	0.8441	0.8499	0.5765

* $\delta_P = [0.2, 0.8]$, $\delta_d = 0.13$, $\epsilon_d = 0.16$, $\sigma = 0.01$, $t = 2$
** $\delta_P = [0.2, 0.8]$, $\delta_d = 0.07$, $\epsilon_d = 0.18$, $\sigma = 0.01$, $t = 2$
*** $\Delta_- = 0.09$, $\Delta_+ = 0.1$, $N_c = 4$, $m = 3$, $t = 2$
**** $\beta = 0.0125$, $\alpha = 7.5$, $\delta = 0.016$, $K = 4$

your attention to the fact that it significantly outperforms both SFCM and LDA in clustering multi-labeled data points. Note that SDC consistently outperforms LDA, which is a specialized algorithm for text clustering. Clearly, in the grid search we identified a number of parameter settings which did not lead SDC to a reasonable performance. This is a normal situation for any algorithm and some recommendations for parameter settings are therefore required to bring the algorithm to or close to its (near) optimal performance.

Optimal ϵ_d and δ_d values can be chosen based on dynamics of group dependence curves (group dependence divided by $\sum_{g_{i,j}>0} G$ as a function of iteration number, see Section 2.3). The curves with higher group dependence values are more likely corresponding to the better parameter values for ϵ_d. The suitable value for δ_d can be selected based on observing the region of group dependence curve where it transitions from being steep to being flat(ter). The point where such transition happens corresponds to the iteration where further division should be stopped since any subsequent dependence gain will be insignificant and produced clusters become less contrastive. Once the point of deflection/stop is decided upon, δ_d is assigned a value that is greater than the dependence gain following next division and smaller than the dependence gain from the past divisions.

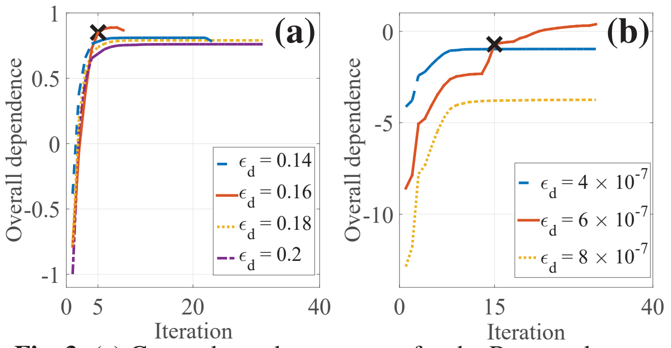


Fig. 3. (a) Group dependence curves for the Reuters data set, (b) group dependence curves for the Genbase data set. The cross signs mark iterations where divisions stop.

Figure 3(a) shows group dependence curves for the Reuters data set for a number of values of ϵ_d. The solid line displays the curve corresponding to $\epsilon_d = 0.16$ resulted in the best average F_1-score, while the dotted line represents the curve corresponding to $\epsilon_d = 0.18$ leading to best NMI score. We can see that the maximal group dependence value is reached by the curve corresponding to $\epsilon_d = 0.16$. Moreover, for this curve the suggested value of δ_d should be approximately 0.13 which agrees with Table 2. Analysis of Figure 3(a) suggests that group dependence curves correlate better with average F_1-score compared to NMI.

3.4. Genbase data set

Genbase data set [17] contains descriptions of patterns (motifs) associated with the most important protein families. It consists of 662 instances with 1185 attributes and each protein can be associated with 27 labels at maximum. All attributes are binary, hence, enabling application of LDA. The data set used here was obtained from the KEEL data repository [18].

To obtain the similarity matrix from the original feature space of the data set, we first constructed a matrix of pair-wise Hamming distances between instances of the data set H $= \{h_{ij} | i, j = 1, .., N\}$ as percentages of the number of coordinates that differ. Then, the similarity matrix S $= \{s_{ij} | i, j = 1, ..., N\}$ was derived from the distance matrix using the following transformation: $s_{ij} = |h_{ij} - \max_{i,j}\ h_{ij}|$.

The testing procedure was similar to the one described in Section 3.3. We compared performance of SDC against SFCM and LDA. The best parameter values were selected based on the results of a grid search (see Table 1 for the parameter grid settings). Performance results are shown in Table 3 where the highest values of the performance measures are marked in bold. As one can notice, under some settings SDC outperforms SFCM and LDA with regard to all performance measures except for F_{1m} for SFCM where it, however, demonstrates comparable results. Furthermore, SDC significantly outperforms both SFCM and LDA in clustering single-labeled data points. In addition, in Figure 3(b) we plot group

Table 3. Genbase results

	Parameters	NMI	F_1	F_{1s}	F_{1m}
SDC	*	**0.6808**	**0.6671**	**0.5840**	0.6729
SFCM	**	0.6186	0.6047	0.4391	0.6737
	***	0.6247	0.6043	0.4397	**0.6788**
LDA	****	0.5051	0.5542	0.5451	0.6394
	*****	0.5195	0.5284	0.4631	0.6260

* $\delta_P = [0.2, 0.8]$, $\delta_d = 0.06$, $\epsilon_d = 6 \times 10^{-7}$, $\sigma = 0.01$, $t = 4$
** $\Delta_- = 0.01$, $\Delta_+ = 0.04$, $N_c = 12$, $m = 5$, $t = 2$
*** $\Delta_- = 0.035$, $\Delta_+ = 0.1$, $N_c = 12$, $m = 5$, $t = 2$
**** $\beta = 0.02$, $\alpha = 1$, $\delta = 0.3$, $K = 13$
***** $\beta = 0.03$, $\alpha = 2$, $\delta = 0.05$, $K = 16$

dependence curves for a number of parameter values of ϵ_d. From this figure we can see that the curve with the maximal group dependence value corresponds to $\epsilon_d = 6 \times 10^{-7}$ which is the same that led to the maximal average F_1 and NMI scores during clustering via SDC. The recommended value of δ_d for this curve shoud be approximately 0.06.

4. CONCLUSIONS

In this work, we presented Soft Dependence Clustering algorithm for revealing structure in data with overlapping clusters and provided recommendations for parameter settings. This algorithm specifically designed for discovering cluster overlaps is an adaptation of Dependence Clustering method introduced earlier. It has a solid probabilistic interpretation derived from Markov chain random-walk model. The method provides firmly accurate results compared to SFCM and LDA used for performance comparison analysis in the presented work. Depending on the data set, SDC provides either significantly higher or comparable accuracy in all performance measures considered in this paper. The method handles well different types of data from different domains such as biology and text data. For text data sets it proved to be more accurate in detecting multiple groups compared to LDA, which is a method specially designed for clustering text data.

Despite the SDC algorithm demonstrating substantially accurate results, it is not specifically designed for clustering big data. As the next step, we plan to work towards seamless and efficient integration of our proposed algorithm into a modern big data framework.

5. REFERENCES

[1] A.J. Battle, E. Segal, and D. Koller, "Probabilistic discovery of overlapping cell processes and their regulation," in *Eight Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, San Diego, CA, April 2004.

[2] D.D. Lewis, "Reuters-21578 text categorization test collection distribution 1.0," Available at `http://archive.ics.uci.edu/ml/machine-learning-databases/reuters21578-mld/reuters21578.html`, September 1997.

[3] G. Cleuziou, "An extended version of the k-means method for overlapping clustering," in *ICPR*. 2008, pp. 1–4, IEEE Computer Society.

[4] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R.J. Mooney, "Model-based overlapping clustering," in *Proc. 11th ACM SIGKDD*, New York, USA, 2005, KDD '05, pp. 532–537, ACM.

[5] K. Lee, A. Gray, and H. Kim, "Dependence maps, a dimensionality reduction with dependence distance for high-dimensional data," *Data Min. Knowl. Discov.*, vol. 26, no. 3, pp. 512–532, May 2013.

[6] H. Park and K. Lee, "Dependence clustering, a method revealing community structure with group dependence," *Know.-Based Syst.*, vol. 60, pp. 58–72, Apr. 2014.

[7] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[9] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[10] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators," in *in Advances in Neural Information Processing Systems 18*. 2005, pp. 955–962, MIT Press.

[11] The MathWorks Inc., "Fuzzy logic toolbox user's guide," Available at `http://www.mathworks.com/help/pdf_doc/fuzzy/fuzzy.pdf`, March 2014.

[12] M. Steyvers, "Matlab topic modeling toolbox 1.4," Available at `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`, 4 April 2011.

[13] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev. E*, vol. 80, pp. 056117, Nov 2009.

[14] J. Yang and J. Leskovec, "Structure and overlaps of ground-truth communities in networks," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 2, pp. 26:1–26:35, Apr. 2014.

[15] C.J. van Rijsbergen, S.E. Robertson, and M.F. Porter, "New models in probabilistic information retrieval," 1980.

[16] G. W. Snedecor and W. G. Cochran, *Statistical Methods*, Iowa State University Press, eighth edition, 1989.

[17] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," in *Proc. of the 10th PCI*, Berlin, Heidelberg, 2005, PCI'05, pp. 448–456, Springer-Verlag.

[18] J. Alcal-Fdez, A. Fernndez, J. Luengo, J. Derrac, and S. Garca, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

PV

SCALABLE IMPLEMENTATION OF DEPENDENCE CLUSTERING IN APACHE SPARK

by

Elena Ivannikova 2017

Proc. of the Conference on Evolving and Adaptive Intelligent Systems, pp. 1-6

Scalable Implementation of Dependence Clustering in Apache Spark

Elena Ivannikova
Department of Mathematical Information Technology
University of Jyväskylä
POBox 35 (Agora), 40014 Jyväskylä
Email: elena.v.ivannikova@student.jyu.fi

Abstract—**This article proposes a scalable version of the Dependence Clustering algorithm which belongs to the class of spectral clustering methods. The method is implemented in Apache Spark using GraphX API primitives. Moreover, a fast approximate diffusion procedure that enables algorithms of spectral clustering type in Spark environment is introduced. In addition, the proposed algorithm is benchmarked against Spectral clustering. Results of applying the method to real-life data allow concluding that the implementation scales well, yet demonstrating good performance for densely connected graphs.**

I. Introduction

Efficient analysis and processing of large-scale datasets requires scalable algorithms and computational frameworks. To address this challenge, distributed large-scale data processing frameworks have emerged in recent years. One of the most widely-used frameworks/platforms for processing large-scale data, Hadoop [1] is an open source implementation of MapReduce [2]. Despite performing relatively well for offline data, it handles real-time stream data poorly. Moreover, Hadoop normally processes data from the disk which is inefficient for data mining applications that often require numerous iterations. It has been reported by a number of works that Hadoop-run algorithms sustain significant performance loss due to disk I/O operations and network communications [3], [4].

Apache Spark [5] is a more recent open-source distributed framework for data analytics which enables, among other things, fast and efficient processing of large streams of data. The key features of Spark are in-memory computations and fault-tolerance. Spark adopts Resilient Distributed Dataset (RDD) [5], a distributed memory abstraction which supports two types of operations: transformations and actions. Transformations define a new RDD based on the existing one, and actions either return a value to the driver program or export data to a persistent storage. When a transformation is executed a new RDD is created with its records distributed across the main memory. An action operation causes each node to process its local set of records and return the result. Spark also supports in-memory caching of datasets which prevents slow disk reads and performs much faster compared to Hadoop-like systems.

Spark proved itself to be effective for performing machine learning and data mining tasks involving big datasets. Recently, many works refer to Spark as a tool for creating competent solutions for data analysis. Many clustering and anomaly detection algorithms have been developed or adapted to Spark due to its efficience and high performance. Apache Spark MLlib library [6] has a number of implemented clustering algorithms such as k-Means, bisecting k-Means, Gaussian mixtures (GMM), and Power Iteration Clustering (PIC). Some other famous clustering algorithms have been implemented in Spark framework but they do not belong to core Spark libraries, e.g. CURE [7] and a scalable random sampling variation of fuzzy c-Means [8].

Spectral clustering methods [9] detect structure of data distribution based on information aquired from the spectrum of the data affinity matrix. One of the hardest computational tasks usually seen in spectral clustering methods is the necessity to solve an eigenvalue problem of the Laplacian matrix derived from the data affinity matrix. Therefore, spectral clustering algorithms often need to be adapted when applied to large-scale datasets. One way to improve efficience is reducing size of the affinity matrix. Another solution is avoiding recomputation as new data points arrive, making the method suitable for real-time algorithms [10], [11]. A number of methods have been developed in order to avoid high complexity caused by calculating spectrum of the Laplacian matrix and to make spectral clustering applicable for large scale datasets. Among them are approximation methods which perform spectral clustering on either a subset of representative data points or randomly sampled values of the affinity matrix and extend the obtained results to the remaining data points [12], [13], [14], [15].

PIC [16] is an efficient method for clustering data which embeds data points in a low-dimensional subspace derived from the affinity matrix, similarly to other spectral clustering methods. Under the hood, PIC applies 1D k-Means to a non-converged top ranked eigenvector. In PIC embedding results in approximation of a linear combination of all eigenvectors of a normalized affinity matrix where eigenvalues serve as coefficients of the linear combination. Such approach is efficient compared to traditional spectral clustering due to performing only a small number of matrix-vector multiplications instead of running iterations until convergence.

Dependence Clustering (DEP) [17] is a method which considers geometric structures of data and is based on maximizing the *group dependence* measure. The method assumes that any two nodes in the graph can be connected through Markovian transitions that enables calculation of *dependence distance* [18] between graph nodes in a certain evolution step. The level of connectivity scale in group assignment can be adjusted improving the flexibility in regulating the level of detail. This approach determines the optimal number of clusters while dividing the data into clusters. This is particularly important for exploratory research related to real world applications with an unknown number of clusters beforehand. In this paper, a Spark-based implementation of DEP [1] which allows better performance for analysis of big datasets is introduced.

The main contributions of the paper consist of

1) Introducing and implementing in Apache Spark a fast approximate diffusion that enables spectral clustering type algorithms in Spark environment.
2) Implementing a scalable version of DEP in Apache Spark framework.

The rest of the paper is organized as follows. Section II provides descriptions of the main concepts and methods used in the paper. Sections II-A - II-C explain the proposed DEP clustering algorithm. Meanwhile, Spark implementation is described in Section II-D. Section III is devoted to the experimental results. It introduces evaluation metrics, data sets and results of the performance tests. Finally, Section IV completes the paper with conclusions and discussions.

II. Methods

A. Preliminary concepts

We start with a graph defined by an affinity matrix A [19]. The graph is formed by a set of data points $\Omega = \{x_i | i = 1, ..., N; x_i \in \mathbb{R}^n\}$. Therefore, the matrix A with entries from $\mathbb{R}$ representing pairwise similarities has size $N \times N$. We define a Markov chain on this graph by transforming A to the transition matrix P and the corresponding t-step transition matrix $\mathrm{P}^t : P^t_{i,j} = Pr(X_t = j | X_0 = i)$, where a probability variable X_0 represents the initial state and X_t is a random walk representing a node at the t-th transition. The transformation from A to P is done by scaling rows of A so that elements in each row sum up to one. Assuming that the whole chain is ergodic and all transitions follow the Markovian property we define statistical dependence $D_{i,j,t} = Dep(X_0 = i, X_t = j)$ [18] by the following equation:

$$D_{i,j,t} = \frac{Pr(X_0 = i, X_t = j)}{Pr(X_0 = i)Pr(X_t = j)}. \quad (1)$$

Statistical dependence captures inter-dependency of the node in the initial state and the node at t-th transition. Let us denote a group assignment vector by $\mathbf{s} = [s_1, ..., s_N]$, where decision variable $s_i = 1$ if data point i belongs to group 1 and $s_i = -1$ if it belongs to group 2. Note that in such notation $(s_i s_j + 1)/2$ is 1 if i and j are in the same group and is 0, otherwise. Thus, given s and t, the group dependence for a particular choice of s is defined as follows:

$$D_t = \frac{1}{2} \sum_{x_i, x_j \in \Omega} \big(D_{i,j,t} - d_0\big)(s_i s_j + 1), \quad (2)$$

[1]https://github.com/Korelena/spark

where $D_{i,j,t}$ is defined in (1), $d_0 = 1 + \epsilon_d$ is the baseline dependence level which is usually set to 1 and ϵ_d is a dependence margin parameter. A constant d_0 effectively normalizes the statistical dependence for each pair of points equaling zero when points are considered to be independent. More details about parameter settings and optimization procedure can be found in [17].

Group dependence described above captures aggregate inter-dependency of points within the groups along with considering the geometrical structure of the data. Hence, statistical dependence serves as a measure of closeness between data points.

B. DEP for two groups

We start with describing the DEP algorithm for a simple case of bisecting a graph. An optimal clustering solution can be achieved through maximizing the group dependence measure D_t by varying the group assignment s of all N points. We constrain the norm of s to be equal to one: $||\mathbf{s}||_2 = 1$ assuming for simplicity that $\epsilon_d = 0$. Moreover, we relax the original formulation (2) so that elements of s become real as the actual optimization is carried out in the domain of real numbers $\mathbb{R}$. Then, we obtain a good partition by solving the following maximization problem:

$$\arg\max_{||\mathbf{s}||=1} D_t = \arg\max_{||\mathbf{s}||=1} \frac{1}{2} \sum_{i,j} (D_{i,j,t} - 1)(s_i s_j + 1).$$

It can be shown [17] that the problem above is equivalent to

$$\arg\max_{||\mathbf{s}||=1} \mathbf{s}^T (\mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T)\mathbf{s},$$

where diagonal matrix $\mathrm{B}^{(t)} : B^{(t)}_{j,j} = Pr(X_t = j) = [x_0^T \mathrm{P}^t]_j$, x_0 is the initial probability vector representing prior information related to the initial states, and $\mathbf{1}$ is the all-ones vector. This constrained optimization problem can be solved by using one of the standard eigendecomposition numerical algorithms, e.g. Power iteration or Arnoldi iteration [20], [27].

Finally, division of the data points is made based on the signs of the eigenvector corresponding to the largest positive eigenvalue of G defined by

$$\mathrm{G} = \mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T. \quad (3)$$

The nonexistence of positive eigenvalues means no possible benefits for increasing D_t from further divisions.

C. DEP for multiple groups

To obtain divisions to multiple groups, we apply a standard subsequent division approach [21]. At every step we consider binary divisions of every group already found during the previous iterations, following the DEP algorithm described in Section II-B. Among possible divisions we proceed with the one that results in the maximal increase of group dependence for the whole dataset, effectively performing greedy search.

For the purpose of preventing the algorithm from defining too small or unclear clusters, we introduce a dependence gain parameter $\delta_d \in [0, 1]$. The minimal dependence gain required to split a cluster into two subclusters is calculated as $\Delta_d =$

$\delta_d \sum_{g_{i,j}>0}$ G, where G is defined in (3). Let us denote a set of points that belong to a split candidate cluster by $\Omega_C \subseteq \Omega$. We denote the within-cluster group configurations before and after the division of Ω_C into two subclusters by $\mathbf{s_C}$ and $\mathbf{s'_C}$, respectively. The division into the subclusters proceeds if $D_t(\mathbf{s'_C}) - D_t(\mathbf{s_C}) > N\Delta_d$, where D_t is defined in (2) and N is size of Ω. Note that all elements from s_C are equal to one.

D. Spark implementation

Our implementation is written in Scala and is inspired by the implementation of PIC [16]. In addition, we implemented a Python wrapper. We used GraphX Spark API as a backend to store sparse dependence matrices in a distributed manner and perform computations [22], [23]. GraphX exposes data-parallel and graph-parallel paradigms that allow a versatile set of operations to be done within a single framework.

Consider the data is represented as a graph $G =< V, E >$ where V and E define nodes and weighted edges, correspondingly. We assume that data chunks of order $O(|V|)$ can be stored on a single machine. We also assume that sparsity of data allows redistributing the data across $O(log(|E|)$ machines. Typical real world large scale graphs tend to respect skewed power-law distributions of node degrees [24], [25], [26].

For computing the largest eigenvector of a matrix we used the Power Iteration (PI) method [27]. PI is an iterative method that works as follows. Starting with an arbitrary initial vector $\mathbf{v}_0 \neq 0$ it performs an update $\mathbf{v}_{k+1} = c\mathrm{A}\mathbf{v}_k$, where A is the affinity matrix, $c = \|\mathrm{A}\mathbf{v}_k\|^{-1}$ is a normalizing constant. Due to simplicity of its operations, as only matrix-vector multiplications are performed, the PI method can be used as an integral part of scalable large-scale data analytics solutions.

The proposed implementation of the DEP algorithm is summarized in a pseudo-code below.

```
Require: Normalized graph $G =< V, E >$ cf. (3), $\Delta_d$
 1: Initialize a list of data structures holding information about each group $L = [l_1]$, where the first group includes all nodes $l_1.V = V$ and has group dependence $l_1.D_t = \sum_{e_{i,j} \in E} e_{i,j}$.
 2: while true do
 3:   maxDepGain = 0
 4:   maxGroup = -1
 5:   for all $l_i$ in L do
 6:     Take a subgraph $G' =< l_i.V, E' > \subseteq G$
 7:     Apply PI algorithm to $G'$ and obtain the highest ranked eigenvector s.
 8:     Split nodes of $G'$ into two sets: $V'_1$ for s $< 0$ and $V'_2$ for s $> 0$
 9:     Take two subgraphs of $G'$: $G'_1 =< V'_1, E'_1 >$ and $G'_2 =< V'_2, E'_2 >$
10:     Set $D^1_t = \sum_{e_{i,j} \in E'_1} e_{i,j}$ and $D^2_t = \sum_{e_{i,j} \in E'_2} e_{i,j}$
11:     Compute a dependence gain of this split as depGain = $D^1_t + D^2_t - l_i.D_t$
12:     if depGain > maxDepGain then
13:       maxDepGain = depGain
14:       maxGroup = $i$
15:     end if
16:   end for
17:   if maxDepGain $> N\Delta_d$ then
18:     Initialize entries for the two new subgroups of maxGroup in L
19:   else
20:     return L which contains entries for all found groups
21:   end if
22: end while
```

Another computational problem that we addressed was a diffusion operator [28]. Iterating Markov transition matrix by taking powers of P has an effect of diffusing probability mass from the high potential regions to the potential lower ones. Since a direct multiplication of the large-scale sparse matrices is computationally demanding we approximate the effect of probability diffusion by the locally guided diffusion of affinity in the original affinity space. Namely, we design a procedure of joining disconnected nodes with the high transitive similarities, i.e. on the path $v_i \xrightarrow{e_{i,k}} v_k \xrightarrow{e_{k,j}} v_j$ between disconnected nodes v_i and v_j all the weights of the edges $e_{i,k}$, $e_{k,j}$ and the influx of the node v_k are relatively high.

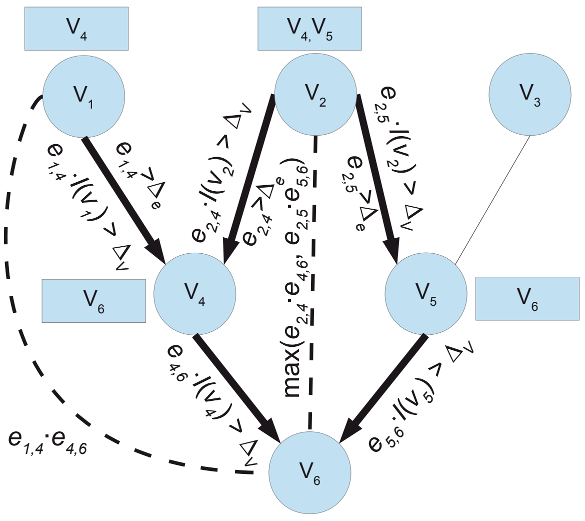

V_4
V_4, V_5
V_1
V_2
V_3
$e_{1,4} > \Delta_e$
$e_{1,4} \cdot I(v_1) > \Delta_v$
$e_{2,4} \cdot I(v_2) > \Delta_v$
$e_{2,4} > \Delta_e$
$\max(e_{2,4} \cdot e_{4,6}, e_{2,5} \cdot e_{5,6})$
$e_{2,5} \cdot I(v_2) > \Delta_v$
$e_{2,5} > \Delta_e$
V_6
V_4
V_5
V_6
$e_{4,6} \cdot I(v_4) > \Delta_v$
$e_{5,6} \cdot I(v_5) > \Delta_v$
$e_{1,4} \cdot e_{4,6}$
V_6

Fig. 1. Illustration of the diffusion procedure. Bold edges denote paths that satisfy constraints for a new edge addition. Expressions along the bold lines are the conditions that enabled emergence of new edges. Added edges are marked dashed. Expressions along the dashed lines determine weights of the newly formed edges. Node names in the boxes denote nodes that are aggregated at a particular node aside.

The procedure goes as follows (see Fig. 1). First, for every node v_i we calculate an influx $I(v_i)$ as a sum of weights of all in-bound edges. Then at every node v_i we aggregate a hash map of nodes v_j such that there exists an outgoing edge from v_i to v_j. IDs of the nodes v_j serve as keys of the hash map and weights of the edges $e_{i,j}$ serve as its values. Entries of only those nodes are aggregated that pass a strength test $e_{i,j} \times I(v_i) > \Delta_v$. At the next step triples of edge, source and destination nodes are considered, i.e. hash maps of two nodes v_i, v_j and their connecting edge $e_{i,j}$ are brought together. A new edge $e_{i,k}$ is added to the graph if $e_{i,j} > \Delta_e$ and there is an entry for the node v_k in the hash map of v_j. The weight of a newly formed edge becomes the maximum aggregation among all the paths that enable the new edge, i.e. $\max_{v_k}(e_{i,k} \times e_{k,j})$. This procedure is guided solely by the local neighbourhood information. Under the assumption of the graph sparsity and with a proper set of parameters it should scale well. Namely, Δ_v along with Δ_e control sizes of the hash maps at each node and the number of added edges. Thus, the procedure effectively simulates diffusion and, at the same time, the graph sparsity remains protected by allowing only strong edges to emerge.

III. Experimental results

A. Experimental environment

In our experiments we used two setups. The local setup is a standalone version of Spark that supports parallelization across multiple cores. The test server had the following characteristics: 1TB of RAM, 64 CPU cores (8 cores Intel Xeon e7-8837 2.67GHz per CPU). For our tests we only reserved 16 cores/executors with 4GB RAM each and 16 GB RAM for the driver program. The second setup is a cluster deployed on Amazon Web Services (AWS) that had 4 slave nodes with 4 cores each and 12.4GB RAM per slave reserved for Spark. A master node had 4 cores and 16GB RAM in total.

B. Evaluation metrics

Given true clustering labels/categories, we used the following two evaluation metrics to measure performance of the algorithms.

Purity [29], [30] focuses on the frequency of the most common category in each cluster and is computed as follows. First, each cluster is assigned to the most frequent category in the cluster. Then the number of correctly assigned items is counted and divided by the total number of clustered items N. the Purity is defined by the following equation:

$$\text{Purity}(\mathbb{C}, \mathbb{B}) = \frac{1}{N} \sum_k \max_j |c_k \cap b_j|,$$

where $\mathbb{C} = \{c_1, c_2, \ldots, c_K\}$ is the set of clusters and $\mathbb{B} = \{b_1, b_2, \ldots, b_J\}$ is the set of categories, c_i is a set of labels assigned to the i-th discovered cluster, and b_j is a set of j-th cluster categories.

Inverse Purity [30] focuses on the frequency of the most common cluster in each category and is defined by the following equation:

$$\text{InversePurity}(\mathbb{C}, \mathbb{B}) = \frac{1}{N} \sum_j \max_k |c_k \cap b_j|.$$

Note, that Purity becomes higher when the number of clusters is large and reaches its maximum when each item gets its own cluster. Inverse Purity reaches its maximum when all items belong to a single cluster. Therefore, a combination of the two measures is normally used for more accurate results.

TABLE I
Diffusion test results reported for the Reuters-1856 dataset.

Parameters				Purity	Inverse Purity
p	t	Δ_e	Δ_v		
1	0	-	-	0.6775	0.8566
0.5	0	-	-	0.2947	1.0
0.5	2	0.3	85.0	0.6439	0.7766
0.5	1	0.3	85.0	0.2947	1.0
0.75	0	-	-	0.2947	1.0
0.75	1	0.3	70.0	0.6210	0.8336

C. Reuters data set

In our tests we refer to the tag *Topics* of the Reuters dataset [31]. We prepared two datasets named Reuters-8852 and Reuters-1876. Reuters-8852 was formed in the following way. Among all topics the following ten were selected: *money-fx*, *grain*, *crude*, *interest*, *trade*, *ship*, *acq*, *earn*, *wheat* and *corn*. Initially, 9400 articles were taken where at least one of the selected topics was present. The affinity matrix for the articles was defined as a cosine similarity among L_2-normalized term frequency (TF) document vectors.

We cleaned the data in the following way. First, we extracted a subset of articles where only one of the selected topics was present. Next, we generated TF features following the procedure of removing punctuation, tokenization and stemming (Porter stemmer [32]). We skipped all the terms that appeared in more than 40% of documents. Among the obtained TF features we selected the 1000 most discriminative features according to the χ^2 test [33], which effectively resulted in a reduced vocabulary of 1000 terms. Similarly, we generated TF matrix for each of the 9400 documents considering only terms from the reduced vocabulary. Finally, we obtained 8852 articles with at least one non-zero feature and built the affinity matrix A which we used for testing the algorithm.

Reuters-1876 was derived in a similar way except that we used 500 TF features and considered only four topics: *grain*, *crude*, *interest* and *trade*. This dataset initially contained 2000 documents before all the empty documents were removed. Finally, Reuters-1876 had 1876 documents.

D. Diffusion test

To verify correctness of the diffusion procedure we run the tests on the Reuters-1876 dataset. Table I displays results of applying DEP to the Reuters-1876 dataset undergoing different subsampling and diffusion rates. Here the parameter p stands for the sampling probability. The parameter t is the number of diffusion iterations applied to a graph. Thus, $t = 0$ means no diffusion was applied. First, we run the DEP algorithm on the original Reuters-1876 dense graph data. Next, we run DEP on the same dataset which was subsampled first by a factor of two ($p = 0.5$) that significantly degraded clustering scores. Applying DEP to the subsampled by a factor of two data with diffusion under parameter settings: t=2, Δ_e=0.3, Δ_v=85.0 produced results comparable to the clustering results without subsampling and diffusion. Moreover, running diffusion under the same parameter settings but only ones ($t = 1$) still resulted in degraded solution. The latter means that subsampling has made severe damage to the graph structure.

We also applied DEP with and without diffusion to a less degraded graph where only quater of the edges were randomly discarded ($p = 0.75$) to verify the effect of the diffusion scale parameter t. In this case running the algorithm without diffusion resulted in much less accurate results compared to the scores of a run with diffusion applied. In all the runs ϵ_d=0.16, δ_d=0.13. These results confirm that the diffusion procedure correctly reconstructs the graph structure.

E. Scalability test

We verify scalability of our implementation using both local and cluster setups. To perform the test we first sparsify the Reuters-8852 dataset by dropping all the edges that have weights less than 0.35. After this step 3742377 edges and all 8852 nodes were retained. Next, we run the DEP algorithm for the successively reduced dataset measuring execution times (see Fig.2). We partitioned the dataset to 16 partitions. In this experiment, other parameters were set to the following values: ϵ_d = 0.35, δ_d = 0.00, t=0.

One can verify that the DEP algorithm scaled near linearly in terms of the number of computations with respect to the number of nodes. From Fig. 2 one can see that the difference in timings between local and cluster setups is nearly constant and is apparently caused by additional network communication in cluster setup.

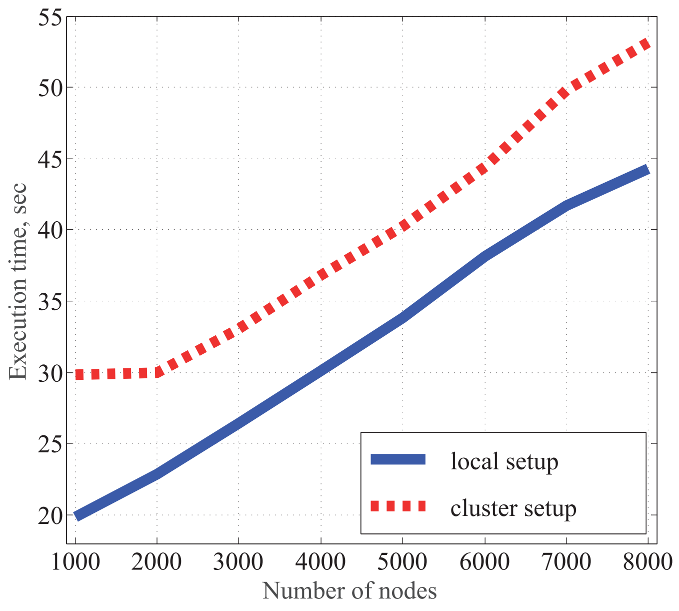


Fig. 2. Execution time of DEP as a function of number of nodes in the dataset.

F. Performance comparison

We compared clustering results of the DEP and Spectral clustering [9] methods applied to the Reuters-1856 dataset.

TABLE II
PERFORMANCE COMPARISON RESULTS REPORTED FOR THE REUTERS-1856 DATASET. MEAN AND (STANDARD DEVIATION) ARE SHOWN FOR PURITY AND INVERSE PURITY.

Method	Purity	Inverse Purity
DEP	**0.6775 (0.0)**	**0.8566 (0.0)**
Spectral clustering	0.3070 (0.0079)	0.6825 (0.2000)

The results are displayed in Table II. For each method we reported mean and standard deviation values of Purity and Inverse Purity computed over 100 iterations. DEP was more accurate compared to Spectral clustering with regard to both measures. Moreover, low standard deviation implies that DEP was more stable compared to Spectral clustering.

IV. CONCLUSION

In this paper we described a scalable Spark-based implementation of the DEP algorithm for clustering data points. The implementation is backed by the efficient Graphx API that supports graph-parallel and data-parallel paradigms. The method belongs to the class of spectral clustering algorithms and performs iteratively greedy binary splits to subgroups, thus, also resembling divisive hierarchical clustering scheme.

We introduced an approximate diffusion algorithm that acts over affinity data matrix and simulates Markov transitions. One should, however, carefully choose parameters in order to avoid memory overflow and to stay in bounded computational resources requirements. An interesting research direction would be to further explore various schemes for carrying out the diffusion.

The tests with real data show that the proposed implementation performs well. Moreover, our algorithm outperformed Spectral clustering which is a common, yet, strong benchmark in cluster analysis. The proposed algorithm can be applied for cluster analysis of large data sets. The potential applications of the algorithm span analysis of text, social networks and network security data, which is a focus of future research.

ACKNOWLEDGMENT

This research was supported by the Nokia Foundation Scholarship funded by Nokia, Finland.

REFERENCES

[1] “Apache hadoop,” Available at http://hadoop.apache.org/.
[2] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” vol. 51, no. 1. New York, NY, USA: ACM, Jan. 2008, pp. 107–113.
[3] K. Kambatla and Y. Chen, “The truth about mapreduce performance on ssds,” in *28th Large Installation System Administration Conference, LISA ’14, Seattle, WA, USA, November 9-14, 2014.*, 2014, pp. 109–118.
[4] F. Pan, Y. Yue, J. Xiong, and D. Hao, “I/o characterization of big data workloads in data centers,” in *Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 4th and 5th Workshops, BPOE 2014, Salt Lake City, USA, March 1, 2014 and Hangzhou, China, September 5, 2014, Revised Selected Papers*, J. Zhan, R. Han, and C. Weng, Eds. Springer International Publishing, 2014, pp. 85–97.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2.

[6] X. Meng, J. K. Bradley, B. Yavuz, E. R. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark," *CoRR*, vol. abs/1505.06807, 2015.

[7] L. Haoxi, D. Min, T. Xue, B. Sheng, C. Dan, and Q. Rongcai, "The spark-based framework for mobile network data and cluster analysis on mobile users' behaviors," in *2015 IEEE International Conference on Robotics and Biomimetics, ROBIO 2015, Zhuhai, China, December 6-9, 2015*, 2015, pp. 487–492.

[8] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy based clustering algorithms to handle big data with implementation on apache spark," in *Second IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2016, Oxford, United Kingdom, March 29 - April 1, 2016*, 2016, pp. 95–104.

[9] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[10] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang, "Incremental spectral clustering by efficiently updating the eigen-system," *Pattern Recognition*, vol. 43, no. 1, 2010.

[11] T. Kong, Y. Tian, and H. Shen, "A fast incremental spectral clustering for large data sets," in *12th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2011, Gwangju, Korea, October 20-22, 2011*, 2011, pp. 1–5.

[12] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 907–916.

[13] L. Wang, C. Leckie, K. Ramamohanarao, and J. C. Bezdek, "Approximate spectral clustering," in *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, 2009, pp. 134–146.

[14] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Jan. 2004.

[15] H. Shinnou and M. Sasaki, "Spectral clustering for a large data set by reducing the similarity matrix size," in *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*, 2008.

[16] F. Lin and W. W. Cohen, "Power iteration clustering," in *ICML*. Omnipress, 2010, pp. 655–662.

[17] H. Park and K. Lee, "Dependence clustering, a method revealing community structure with group dependence," *Know.-Based Syst.*, vol. 60, pp. 58–72, Apr. 2014.

[18] K. Lee, A. Gray, and H. Kim, "Dependence maps, a dimensionality reduction with dependence distance for high-dimensional data," *Data Min. Knowl. Discov.*, vol. 26, no. 3, pp. 512–532, May 2013.

[19] U. Ozertem, D. Erdogmus, and R. Jenssen, "Mean shift spectral clustering," *Pattern Recognition*, vol. 41, no. 6, pp. 1924 – 1938, 2008.

[20] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, ser. Texts in Applied Mathematics. Paris, FR: Springer, 2007.

[21] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[22] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "Graphx: A resilient distributed graph system on spark," in *First International Workshop on Graph Data Management Experiences and Systems*, ser. GRADES '13. New York, NY, USA: ACM, 2013, pp. 2:1–2:6.

[23] R. S. Xin, D. Crankshaw, A. Dave, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "Graphx: Unifying data-parallel and graph-parallel analytics," *CoRR*, vol. abs/1402.2394, 2014. [Online]. Available: http://arxiv.org/abs/1402.2394

[24] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet," *Glottometrics*, vol. 3, pp. 143–150, 2002.

[25] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Comput. Netw.*, vol. 33, no. 1-6, pp. 309–320, Jun. 2000.

[26] S. Lattanzi and Y. Singer, "The power of random neighbors in social networks," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ser. WSDM '15. New York, NY, USA: ACM, 2015, pp. 77–86.

[27] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed. JHU Press, 2013.

[28] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators," in *in Advances in Neural Information Processing Systems 18*. MIT Press, 2005, pp. 955–962.

[29] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[30] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retr.*, vol. 12, no. 4, pp. 461–486, Aug. 2009.

[31] D. Lewis, "Reuters-21578 text categorization test collection distribution 1.0," Available at http://archive.ics.uci.edu/ml/machine-learning-databases/reuters21578-mld/reuters21578.html, 26 September 1997.

[32] C. van Rijsbergen, S. Robertson, and M. Porter, "New models in probabilistic information retrieval," 1980.

[33] G. W. Snedecor and W. G. Cochran, *Statistical Methods*, 8th ed. Iowa State University Press, 1989.

PVI

ANOMALY DETECTION APPROACH TO KEYSTROKE DYNAMICS BASED USER AUTHENTICATION

by

Elena Ivannikova, Gil David, and Timo Hämäläinen 2017

Proc. of the IEEE Symposium on Computers and Communications, pp. 885-889

Anomaly Detection Approach to Keystroke Dynamics Based User Authentication

Elena Ivannikova, Gil David and Timo Hämäläinen
Department of Mathematical Information Technology
University of Jyväskylä
POBox 35 (Agora), 40014 Jyväskylä, Finland
Email: elena.v.ivannikova@student.jyu.fi, gil.david@jyu.fi, timo.hamalainen@jyu.fi

Abstract—**Keystroke dynamics is one of the authentication mechanisms which uses natural typing pattern of a user for identification. In this work, we introduced Dependence Clustering based approach to user authentication using keystroke dynamics. In addition, we applied a k-NN-based approach that demonstrated strong results. Most of the existing approaches use only genuine users data for training and validation. We designed a cross validation procedure with artificially generated impostor samples that improves the learning process yet allows fair comparison to previous works. We evaluated the methods using the CMU keystroke dynamics benchmark dataset. Both proposed approaches outperformed the previous state-of-the-art results for the CMU dataset for unsupervised learning.**

I. Introduction

With the rapidly expanding internet services industry and, thus, the increasing importance of cyber security, the user identification and authentication problems have become a focus for many research labs. User's identity is normally verified by an access control mechanism, which performs the authentication task. Traditional approaches to access control provide good performance, however, they have some limitations. For example, passwords or PINs can be forgotten, lost or stolen which threatens security. Hence, new forms of authentication based on conjunction of traditional methods with biometrics are becoming more popular within computer security.

Biometrics are defined as the physical traits and behavioral characteristics that identify a living person [1]. Among them, keystroke dynamics [2] is considered as a strong behavioral biometric based authentication system. Keystroke dynamics statistics allow extracting timing features containing information about human's typing rhythms, i.e. time intervals between (a) the key presses of consecutive keys, (b) pressing a key and releasing next key, (c) pressing and releasing a key. Such typing patterns can serve as a human's identifier due to their ability to activate similar behavioral and cognitive mechanisms cf. handwritten signatures. Moreover, implementation cost of keystroke dynamics is very low as only a keyboard is needed for data collection. Among other advantages are the ability to operate in hidden mode, high user acceptance and ease of integration to existing security systems [3].

Despite all the advantages keystroke dynamics can be influenced by external factors, e.g. environmental conditions or keyboard device, or emotional state producing some noise and causing lower accuracy and permanence [4], [3]. With the help of statistical, machine learning or other algorithms researchers can identify behavioral patterns which allow distinguishing among the users based on specific characteristics. Analysis of typing patterns can be a powerful security tool for detecting intrusions or threats, or for distinguishing between genuine users and impostors during authentication [5], [6], [7], [8].

The rest of the paper is organized as follows. Section II shortly reviews the current state of keystroke dynamics techniques. Section III briefly describes the dataset used in this work and the data collection procedure. Section IV provides descriptions of the main concepts and methods used in the paper. Sections IV-A - IV-D explain the DC algorithm and the DC based anomaly detection method, while Section IV-E refers to k-NN based anomaly detection approach. Evaluation procedure is described in Section IV-F. Meanwhile, Section V is devoted to the experimental results. It describes parameter estimation procedure and results of the performance tests. Finally, Section VI concludes the paper.

II. Related Work

Most of the previous research works in keystroke dynamics refer to authentication systems. For user's authentication via keystroke dynamics either *static* or *free* text models can be used. The majority of previous works focus on *static* text when users type specific predefined text such as password [7], [2]. In more advanced and secure systems, users are being continually authenticated and monitored based on *free* text models when users type arbitrary input text of any length [9], [10], [2].

A vast amount of performance results obtained using various keystroke dynamics datasets have been reported by studies. Most of the studies collected own data, therefore, making performance comparison among the works difficult. To tackle this issue, in [11] authors present a comprehensive comparative study of detecting anomalies using keystroke dynamics dataset (CMU), thus, making a good benchmark. The authors collected data and implemented 14 anomaly detection algorithms for detecting anomalies in keystroke dynamics. Each detector was trained on a set of timing vectors of a true user, thus, providing a true-user behavioral model. Then the rest of data samples were tested against the true-user behavioral model and assigned an anomaly score. The parameters tuning was not performed in the experiments of this study due to a possible bias in the evaluation results. Instead, the authors used the parameters reported in the source studies. In addition to [11] there are a number of works presenting performance results for the CMU keystroke dynamics benchmark dataset. According to [11] the top detectors demonstrating best results on the CMU dataset are the scaled Manhattan distance [12] and the nearest neighbor with Mahalanobis distance [13] with equal error rate (EER) values of 0.096 and 0.100, correspondingly. The best zero-miss false-alarm (ZMFAR) rate belongs to the nearest neighbor detector using Mahalanobis distance with the value of 0.482. Classical Mahalanobis [14] and the normed Mahalanobis [15] detectors demonstrate equal ZMFAR values of 0.482.

Following [11] a number of works proposed new algorithms and compared their performance with existing results. In [16], the authors followed the study in [11] by introducing new detectors and improving the results by using the same protocol and evaluation procedure in order to guarantee fair performance comparison. They introduced a new distance measure by combining Mahalanobis and Manhattan measures and used it in combination with the nearest neighbor classifier. They improved EER by 0.9% and ZMFAR by 4.5% compared to the best results in [11]. Furthermore, the Gaussian mixture model (GMM) was applied [17] producing EER of 0.087. Despite the authors also reported additional even better results we omit them here as the testing procedure was different from the one described in [11] making the fair comparison impossible. Another work [18] devoted to applications of neural networks to keystroke data reports improved results compared to the best performance values from [11]. The authors got EER of 0.0773 by using Levenberge-Marquardt backpropagation network. However, they incorporated negative examples into the training set. This prevents fair comparison with [11] where detectors were trained with the use of only positive samples. All aforementioned performance results can be found in Table I.

In this work, we propose two anomaly detection approaches based on *k*-nearest neighbors (*k*-NN) and dependence clustering (DC) [19]. Despite its simplicity, k-NN is a strong benchmark and often provides state-of-the art results in different tasks including biometric identification and authentification [20]. In our study, we employ a k-NN based approach combining with Manhattan distance. DC is a spectral clustering type algorithm which has been used for the clustering tasks. In this study, we adapt DC to solving anomaly detection problems. We test the methods on the CMU dataset and compare the obtained results with the performance reported in [11]. We reproduce training and evaluation procedures according to [11] to ensure fair comparison among detectors. Both proposed methods outperform the previous known state-of-the-art results on the CMU dataset.

III. Data

Detailed information of how the data was collected can be found in [11]. In this section, we provide a brief description of the data and the data collection procedure. For password generation and verifying its strength, publicly available tools were used [21], [22].

The password was generated as a sequence of 10 characters containing letters, numbers and punctuation signs. The obtained sequence was manually modified by altering some punctuation and casing in order to better meet requirements of a strong password. This resulted in password *.tie5Roanl* that still was rated strong. During data collection the same password was typed by all subjects.

The data was collected from 51 subjects of different age, sex and handedness groups. Each subject resulted in 400 password-typing samples. After all data had been collected, a set of timing features were extracted from raw data. Finally, 31 timing features were generated. Despite known correlations and linear dependency among timing features all of them were left in the data with a purpose of being useful in evaluation of future works. Possible adverse effect on some detectors can be avoided through a careful feature selection procedure [23].

IV. Methods

A. Preliminary concepts

Dependence Clustering (DC) is a method which considers geometric structures of data and is based on maximizing the group dependence measure. First, let us introduce essential assumptions regarding the data and concepts foundational this algorithm.

Given a set of data points $\Omega = \{x_i | i = 1, ..., N; x_i \in \mathbb{R}^n\}$ we form a graph defined by a similarity matrix S of size $N \times N$. By scaling rows of S so that elements in each row sum up to one we transform S to the transition matrix P and the corresponding t-step transition matrix P^t, thus, defining the Markov chain on this graph. The t-step transition matrix is calculated as $\mathrm{P}^t : P^t_{i,j} = Pr(X_t = j | X_0 = i)$, where X_0 represents probability at the initial state and X_t is a random walk representing a node at the t-th transition. Further, we assume that the whole chain is ergodic and that any two nodes in the graph can be connected through Markovian transitions. This enables calculation of statistical dependence between graph nodes in a certain evolution step $D_{i,j,t} = Dep(X_0 = i, X_t = j)$ [24] that is defined by the following equation:

$$D_{i,j,t} = \frac{Pr(X_0 = i, X_t = j)}{Pr(X_0 = i)Pr(X_t = j)}. \tag{1}$$

Statistical dependence serves as a measure of closeness between data points.

Then we define group dependence D_t as

$$D_t(\mathrm{s}) = \sum_{x_i, x_j \in \Omega} \big(D_{i,j,t} - d_0\big)(s_i s_j + 1)/2, \tag{2}$$

where $D_{i,j,t}$ is defined in (1), $\mathrm{s} = [s_1, ..., s_N]$ is a group assignment vector, where decision variable $s_i = 1$ if data point i belongs to group 1 and $s_i = -1$ if it belongs to group 2 and $d_0 = 1 + \epsilon_d$ is the baseline dependence level which is usually set to 1. More detailed information about parameter settings and optimization procedure can be found in [19].

B. DC for two groups

The actual optimization is carried out in the domain of real numbers $\mathbb{R}$. Hence, we relax the original formulation (2) so that elements of $\mathbf{s}$ become real. We start with a simple case of bisecting a graph. By varying the group assignment $\mathbf{s}$ of all N points and constraining the L_2 norm of $\mathbf{s}$ to be equal to one: $||\mathbf{s}||_2 = 1$ we obtain a good partition through maximizing the group dependence measure $D_t(\mathbf{s})$ as follows:

$$\begin{aligned}\arg\max_{||\mathbf{s}||=1} D_t(\mathbf{s}) &= \arg\max_{||\mathbf{s}||=1} \sum_{i,j} (D_{i,j,t} - 1)(s_i s_j + 1)/2 \\ &= \arg\max_{||\mathbf{s}||=1} \mathbf{s}^T (\mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T)\mathbf{s} \qquad (3)\end{aligned}$$

where $\mathrm{B}^{(t)} : B^{(t)}_{j,j} = Pr(X_t = j) = [x_0^T \mathrm{P}^t]_j$, x_0 is the initial probability vector representing prior information related to the initial states, ϵ_d is assumed to be 0, for simplicity. The division of the data points is made based on the signs of the eigenvector corresponding to the largest positive eigenvalue of G_0 and stops when we get no positive eigenvalues. The matrix G_0 is defined by

$$\mathrm{G}_0 = \mathrm{P}^t (\mathrm{B}^{(t)})^{-1} - \mathbf{1}\mathbf{1}^T. \qquad (4)$$

Note that generally matrix G_0 is not symmetric. However, the nature of many datasets, including the CMU dataset used in this study, implies symmetry of similarity relation which obeys commutative property. Therefore, in this paper we make divisions based on eigendecomposition of a symmetric matrix $\mathrm{G} = \mathrm{G}_0 + \mathrm{G}_0^{\mathrm{T}}$.

C. DC for multiple groups

In order to obtain divisions to multiple groups a standard subsequent division approach is applied [25]. At every step we make binary splits of each group already found during the previous iterations, following the procedure described in Section IV-B. We proceed with an optimal division as the one which resulted in the maximal increase of group dependence for the whole dataset. Moreover, the following condition must fulfill: $D_t(\mathrm{sgn}(\mathbf{s}'_C)) - D_t(\mathbf{s}_C) > N\Delta_d$, where $\mathbf{s}_C$ and $\mathbf{s}'_C$ are within-cluster group configurations of a split candidate cluster $\Omega_C \subseteq \Omega$ before and after the division, correspondingly, $D_t(\mathbf{s})$ is defined by (2) and N is the size of Ω_C. $\Delta_d = \delta_d \sum_{g_{i,j}>0} \mathrm{G}$ is the minimal dependence gain required to split a cluster into two sub-clusters where $\delta_d \in [0, 1]$ is a dependence gain parameter. δ_d serves as a regularization parameter which prevents the algorithm from defining too small or too unclear clusters. Figure 1 displays pseudo code summarizing the above steps.

D. DC-based anomaly detection

For discovering anomalies we use the following procedure. First, we apply z-score normalization [26] to the data. Next, we build a similarity matrix by first computing pairwise distance matrix using either Manhattan or Euclidean distance. We transform the distance matrix to the similarity matrix by using the following non-linearity $s_{ij} = \exp(-\alpha d_{ij})$, where d_{ij} denotes an element of the pairwise distance matrix, s_{ij} denotes

```
Require: Set of data points $\Omega$, similarity matrix S, parameters $t$, $\delta_d$, $\epsilon_d$.
  Initialize a set $\Psi = \{\Omega\}$, $\Delta_d$ as described in Section IV-C;
  Compute transition matrix P from S (see Section IV-A);
  while true do
    Set $\Omega'_C = \emptyset$, $\Omega' = \emptyset$;
    for all $\Omega_C$ in $\Psi$ do
      Find a sub-division $\mathbf{s}'_C$ of $\Omega_C$ by solving (3);
      Define $\Omega^1_C = \{x_i | \mathrm{sgn}(\mathbf{s}'_C(i)) > 0\}$ and
             $\Omega^2_C = \{x_i | \mathrm{sgn}(\mathbf{s}'_C(i)) \leq 0\}$;
      if $D_t(\Omega^1_C) + D_t(\Omega^2_C) - D_t(\Omega_C) > N\Delta_d$ then
        $\Omega'_C = \Omega_C$, $\Omega' = \{\Omega^1_C, \Omega^2_C\}$;
      end if
    end for
    if $\Omega'_C \neq \emptyset$ then
      Apply the sub-division of $\Omega'_C$ by replacing $\Omega'_C$ in $\Psi$ by the two elements of the set $\Omega'$;
    else
      Finish the loop.
    end if
  end while
```

Fig. 1. Pseudo code for DC algorithm

an element of the similarity matrix and α is a scale parameter. We anticipate that this non-linearity matches distribution of the pairwise distances well. In other words, it gives more resolution in the region of the distances we are interested in the most. Further, using the DC algorithm we divide the training set into clusters. The median Manhattan distance between a test sample and each cluster mean is then used as anomaly score for this test sample.

E. k-NN-based anomaly detection

k-NN [14] is a well-known method used for classification or regression. It belongs to the category of instance-based learning methods when training phase is essentially missing. All computations are done locally with k nearest neighbors from the training set during testing phase.

For discovering anomalies we use the following procedure. First, we apply z-score normalization [26] to the data. Then for every test sample we compute Manhattan distance between itself and the mean value of its k-nearest neighbors. This distance is used as anomaly score for the test sample.

F. Evaluation

In our experiments we used the same procedure of building training and testing sets as described in [11]. Thus, for each user the first 200 samples were assigned to the training set and the last 200 samples were assigned to the test set as positive samples (true user's patterns). Moreover, the first 5 repetitions from every other user were added to the test set as negative samples (impostor's patterns). Note that training sets are constructed so that they contain only positive examples. To be able to perform cross-validation we need, however, negative samples as well. Using the true impostors' patterns during cross-validation would lead to unfair comparison with the results from [11] as our algorithms will indirectly learn to discriminate among positive and negative samples. Therefore, we

TABLE I
PERFORMANCE COMPARISON OF THE BEST REPORTED ALGORITHMS ON THE SAME CMU DATASET. MEAN AND (STANDARD DEVIATION) ARE SHOWN FOR THE EQUAL ERROR RATE (EER) AND ZERO-MISS FALSE-ALARM RATES (ZMFAR).

Algorithm	Parameters	EER	ZMFAR
Nearest Neighbor (Mahalanobis)		0.100 (0.064)	0.468 (0.272)
Manhattan (scaled)		0.096 (0.069)	0.601 (0.337)
GMM		0.087 (0.058)	-
Combined Mahalanobis and Mahattan distance		0.084 (0.056)	-
k-NN (Manhattan)	k=3	0.078 (0.055)	0.385 (0.267)
k-NN (Manhattan)	k=6	0.078 (0.056)	0.377 (0.272)
k-NN (Manhattan)	k=8	0.078 (0.057)	0.377 (0.276)
DC (Euclidean)	*	0.078 (0.056)	0.390 (0.280)
DC (Manhattan)	**	**0.077 (0.055)**	**0.358 (0.256)**

* $\delta_d = 0$, $\epsilon_d = 0.001$, $\alpha = 3.33$, $t = 1$, Manhattan distance; ** $\delta_d = 0$, $\epsilon_d = 0.01$, $\alpha = 10$, $t = 1$, Manhattan distance

designed a procedure to generate artificial negative samples. For each user we generated 200 additional negative samples by randomly sampling values for every feature independently from the entire data. Thus, for every user we obtained 200 new artificial samples served as impostors. Finally, we merged the obtained negative samples with the original training sets. We ran 10-fold cross validation procedure on the obtained validation sets of 400 samples independently for each user. Note that the randomly obtained artificial samples are not the points from the original dataset. Therefore, with this procedure we avoid using real test samples in the training/validation phase.

After the optimal parameter values had been set up the methods were red trained and tested using the same procedure as described in [11]. The obtained anomaly scores were converted into standard measures of error. The following two measures have been used for estimating performance of all detectors.

- **Equal error rate** (EER) [27] corresponds to a point on the ROC curve where miss rate and false-alarm rate are equal.
- **Zero-miss false-alarm rate** (ZMFAR) [13] is a measure minimizing false-alarm rate constraining the miss rate is zero.

V. RESULTS

The results corresponding to the best parameter values obtained during cross validation as well as the parameter values themselves are shown in Table I. For reference, Table I also displays the best results from the previous studies related to the CMU dataset, reported in Section II. One can see that the presented k-NN and DC approaches demonstrate improved results compared to the previous works. Thus, EER was improved by 7% compared to combined Mahalanobis and Manhattan distance detector [16]. We cannot compare our ZMFAR with the results from [16] as it was not reported in their work. Although, we improved ZMFAR by 11% compared to Nearest Neighbor (Mahalanobis) demonstrating the lowest ZMFAR in [11].

The best results have been shown by the DC method when using Manhattan distance in both stages of the algorithm: generating similarity matrix at the step of clustering and calculating distance at the step of discovering anomalies. We pay attention to the fact that not only the EER and ZMFAR performance measures have been improved but also their standard deviation values (displayed in brackets) became lower, which signifies superior robustness of the proposed methods.

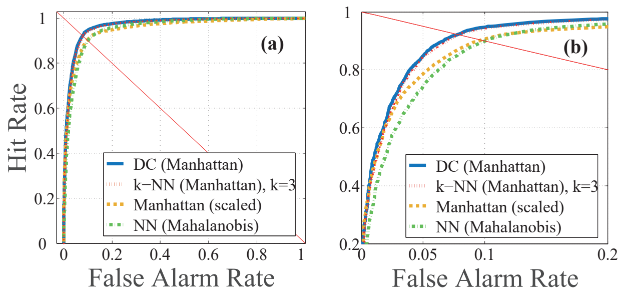


Fig. 2. (a) - ROC curves, (b) - zoomed in ROC curves for anomaly detection algorithms.

To visualize the results we plot ROC curves in Figure 2. Figure 2(b) is a zoomed version of Figure 2(a). ROC curves corresponding to the DC and k-NN based approaches proposed in this paper are displayed by plain and dotted lines, correspondingly. Furthermore, by dashed and dash-dot lines we plot ROC curves for Manhattan (Scaled) and Nearest Neighbor (Mahalanobis) detectors which demonstrated the best EER and ZMFAR in [11]. The threshold for calculating EER is chosen so that detector miss and false-alarm rates are equal. In Figure 2 it corresponds to the point where the line $y = 1 - x$ displayed in red crosses the detector ROC curve.

While computing ROC curves we used concatenated prediction results for all users, i.e. the varied parameter for building ROC curves represented global user-wide threshold (anomaly score) values. Note, this approach is different from the one used for calculation of EER and ZMFAR scores in Table I as the scores were taken as averages from per-user ROC curves. In the latter case the ROC curve parameter was varied for each user independently representing individual user threshold. The obtained ROC curve is a smoothed version of the individual user ROC curves due to substantial variation in ZMFAR rate, in particular. Average EER and ZMFAR rates are higher than the ones measured from the user-wide ROC curve, since by varying anomaly score individually for every user we have more degrees of freedom, i.e. more powerful model. The ROC curves show that the methods generalize well even using global anomaly score threshold and can be further improved by choosing specialized per-user thresholds.

VI. Conclusions

In this work we proposed two approaches for detecting anomalies in the CMU dataset. One approach is based on the well-known k-NN and the other approach is based on DC thoroughly described here. Both proposed approaches outperform previous results that we know for this dataset respecting unsupervised learning setting. The main contributions of our work include:

(a) We improved upon the previous state-of-the-art results for the CMU dataset reported in [11], [16] for the unsupervised learning.

(b) We designed a cross validation procedure with artificially generated negative samples that allows avoiding the use of true negative samples in the learning process. Using the true negative samples during cross-validation would prevent fair comparison with the previous studies and result in violation of purely unsupervised learning setting. The improved results for the CMU dataset, indirectly, justify the validity of this procedure.

(c) We adapted a spectral clustering style algorithm previously used only for clustering problems to anomaly detection.

The practical implications of the presented results manifest in enabling more accurate and robust intrusion detection systems.

In the future the proposed approaches can be extended to other datasets related to more diverse security threats. Moreover, we plan to extend the DC approach to using dependence distance when computing anomaly score or determining the closest cluster of a test point. This extension requires changes in implementation of the DC algorithm that will allow performing real-time incremental computations with test points.

References

[1] J. Poss, D. Boye, and M. Mobley, "Biometric voice authentication," Jun. 10 2008, uS Patent 7,386,448. [Online]. Available: https://www.google.ch/patents/US7386448

[2] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation Computer Systems*, vol. 16, no. 4, pp. 351–359, 2000.

[3] P. S. Teh, A. B. Jin Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Scientific World Journal*, vol. 2013, p. 24 pages, 2013.

[4] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 715–724.

[5] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Commun. ACM*, vol. 33, no. 2, pp. 168–176, Feb. 1990.

[6] J. Ho and D. Kang, "Sequence alignment with dynamic divisor generation for keystroke dynamics based user authentication," *J. Sensors*, vol. 2015, pp. 935 986:1–935 986:14, 2015.

[7] F. Bergadano, D. Gunetti, and C. Picardi, "Identity verification through dynamic keystroke analysis," *Intell. Data Anal.*, vol. 7, no. 5, pp. 469–496, Oct. 2003.

[8] A. Messerman, T. Mustafic, S. A. Çamtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *2011 IEEE International Joint Conference on Biometrics, IJCB 2011, Washington, DC, USA, October 11-13, 2011*, 2011, pp. 1–8.

[9] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[10] P. Kang and S. Cho, "Keystroke dynamics-based user authentication using long and free text strings from various input devices," *Information Sciences*, vol. 308, pp. 72 – 93, 2015.

[11] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics." in *DSN*. IEEE Computer Society, 2009, pp. 125–134.

[12] L. C. F. Araújo, L. H. R. Sucupira, M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti, *User Authentication through Typing Biometrics Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 694–700.

[13] S. Cho, C. Han, D. H. Han, and H. il Kim, "Web based keystroke dynamics identity verification using neural network," *Journal of Organizational Computing and Electronic Commerce*, vol. 10, pp. 295–307, 2000.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[15] S. Bleha, C. Slivinsky, and B. Hussien, "Computer-access security systems using keystroke dynamics." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 12, pp. 1217–1222, 1990.

[16] Y. Zhong, Y. Deng, and A. K. Jain, "Keystroke dynamics for user authentication." in *CVPR Workshops*. IEEE Computer Society, 2012, pp. 117–123.

[17] Y. Deng and Y. Zhong, "Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets." *ISRN Signal Processing*, vol. 2013, p. 7 pages, 2013.

[18] Y. Uzun and K. Bicakci, "A second look at the performance of neural networks for keystroke dynamics using a publicly available dataset," *Comput. Secur.*, vol. 31, no. 5, pp. 717–726, Jul. 2012.

[19] H. Park and K. Lee, "Dependence clustering, a method revealing community structure with group dependence," *Know.-Based Syst.*, vol. 60, pp. 58–72, Apr. 2014.

[20] J. Hu, D. Gingrich, and A. Sentosa, "A k-nearest neighbor approach for user authentication through biometric keystroke dynamics." in *ICC*. IEEE, 2008, pp. 1556–1560.

[21] P. Tools, "Security guide for windowsrandom password generator," http://www.pctools.com/guides/password/, 2008.

[22] Microsoft, "Password checker," Available at http://www.microsoft.com/protect/yourself/password/checker.mspx., 2008.

[23] E. Yu and S. Cho, "Ga-svm wrapper approach for feature subset selection in keystroke dynamics identity verification," in *Proceedings of the International Joint Conference on Neural Networks*. IEEE Press, 2003, pp. 2253–2257.

[24] K. Lee, A. Gray, and H. Kim, "Dependence maps, a dimensionality reduction with dependence distance for high-dimensional data," *Data Min. Knowl. Discov.*, vol. 26, no. 3, pp. 512–532, May 2013.

[25] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[26] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The elements of statistical learning : data mining, inference, and prediction*, ser. Springer series in statistics. New York: Springer, 2009.

[27] P. Kang, S.-s. Hwang, and S. Cho, "Continual retraining of keystroke dynamics based authenticator," in *Proceedings of the 2007 International Conference on Advances in Biometrics*, ser. ICB'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 1203–1211.

PVII

PROBABILISTIC TRANSITION-BASED APPROACH FOR DETECTING APPLICATION-LAYER DDOS ATTACKS IN ENCRYPTED SOFTWARE-DEFINED NETWORKS

by

Elena Ivannikova, Mikhail Zolotukhin, and Timo Hämäläinen 2017

Proc. of the 11th International Conference on Network and System Security, pp. 531-543

Probabilistic Transition-Based Approach for Detecting Application-Layer DDoS Attacks in Encrypted Software-Defined Networks

Elena Ivannikova( ), Mikhail Zolotukhin, and Timo Hämäläinen

Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, (Agora), 40014 Jyväskylä, Finland
{elena.v.ivannikova,mikhail.zolotukhin,timo.hamalainen}@jyu.fi

Abstract. With the emergence of cloud computing, many attacks, including Distributed Denial-of-Service (DDoS) attacks, have changed their direction towards cloud environment. In particular, DDoS attacks have changed in scale, methods, and targets and become more complex by using advantages provided by cloud computing. Modern cloud computing environments can benefit from moving towards Software-Defined Networking (SDN) technology, which allows network engineers and administrators to respond quickly to the changing business requirements. In this paper, we propose an approach for detecting application-layer DDoS attacks in cloud environment with SDN. The algorithm is applied to statistics extracted from network flows and, therefore, is suitable for detecting attacks that utilize encrypted protocols. The proposed detection approach is comprised of the extraction of normal user behavior patterns and detection of anomalies that significantly deviate from these patterns. The algorithm is evaluated using DDoS detection system prototype. Simulation results show that intermediate application-layer DDoS attacks can be properly detected, while the number of false alarms remains low.

Keywords: DDoS attack · Anomaly detection · SDN · Clustering · Behavior pattern · Probabilistic model

1 Introduction

Distributed Denial-of-Service (DDoS) is a coordinated attack which by using multiple hosts prevents legitimate users from accessing a specific network resource, e.g. email, websites, online banking, etc. In DDoS attack, by taking control of the computer and sending a stream of packets an attacker may perform attacks to other computers by sending spam messages or huge amount of data to a website. The target server, overloaded with requests, either becomes very slow even unusable or totally crashes since it can only process a certain number of requests at once. Thus, the server becomes unavailable to the legitimate clients. Another way of the attack is sending malformed packets that cause

Z. Yan et al. (Eds.): NSS 2017, LNCS 10394, pp. 531–543, 2017.
DOI: 10.1007/978-3-319-64701-2_40

the target machine to freeze or reboot [15]. There are many other ways to deny services on the Internet [21]. DDoS attacks have become a major threat to the stability in modern high-speed networks [19]. Being hard to detect and abort in a timely fashion, these attacks can be used to disable strategic business, government, media and public utility sites prompting victims to loose productivity, revenue and reputation.

Traditional DDoS attacks are carried out at the network layer. Among them are volume-based attacks (e.g. UDP floods, ICMP floods, etc.) and protocol attacks (e.g. SYN floods, Smurf DDoS, etc.). Volume-based attacks attempt to consume the bandwidth either within the target network/service, or between the target network/service and the rest of the Internet, when protocol attacks attempt to consume actual server or intermediate communication equipment resources, such as firewalls and load balancer. Recently, these types of attacks have been well studied and various schemes for protecting network against such attacks have been reported [2,8,14]. Application-layer attack is a more advanced attack which targets vulnerabilities in operative systems and web applications. These attacks can be performed by seemingly innocent and legitimate requests from only a few attacking machines generating low traffic rate, which makes them difficult to detect and mitigate.

One of the most frequent application-layer DDoS attacks nowadays are attacks that involve the use of HTTP protocol. These attacks can be grouped into three major categories, depending on the level of sophistication [21]. *Trivial attacks*, where each bot sends a limited number of unrelated HTTP attacks towards the target site, comprise the majority of application-level DDoS attacks on the Internet. In *intermediate attacks* bots continuously generate random sequences of browser-like requests of web pages with all embedded content. Such procedure allows the attack traffic fitting better in regular human requests. *Advanced attacks* consist of a carefully chosen sequence of HTTP requests in order to better mimic the browsing behavior of regular human visitors. Advanced DDoS attacks are believed to raise popularity in the future [21].

Defending against a trivial HTTP attack does not require a complex detection system. Trivial attack can be detected by inspecting each request to determine if it comes from a legitimate user. Intermediate and advanced attacks, however, require more sophisticated techniques [21]. To name a few, paper [24] analyses intermediate application-layer DDoS attacks by defining a model of normal user behavior via a number of clustering techniques and comparing conversations against such normal patterns. Xu *et al.* [23] model user browsing behavior by random walk graph and identify attackers based on analysis of their page-request sequences. Paper [3] proposes a new clustering algorithm against HTTP-GET attacks using entropy-based clustering and Bayes factor analysis for classification of legitimate sessions. Most of the current studies devoted to HTTP-based DDoS attack detection focus on un-encrypted HTTP traffic. Nowadays many DDoS attacks are utilizing secure protocols for data encryption in the application layer of network connections making their detection more difficult. In this work, we concentrate on intermediate attacks in encrypted traffic.

Recently, cloud computing has become a strong contender to traditional on-premise implementations. The main reason is that cloud environments offer advantages such as on-demand resource availability, pay as you go billing, better hardware utilization, no in-house depreciation losses, and, no maintenance overhead [20]. Cloud resources are provided to the customers in the form of virtual machines (VMs). Cloud service provider has to guarantee the security of the machines by filtering unwanted traffic from other cloud customer networks and external hosts. Despite willing to be secured against attacks, cloud customers may wish to remain their traffic un-encrypted. Thus, the cloud service provider has to detect attacks without relying on encrypted packet payload. With the emergence of cloud computing, many attacks, including DDoS attacks, have changed their direction towards cloud environment. In particular, DDoS attacks have changed in scale, methods, and targets and become more complex by using advantages provided by cloud computing.

Modern cloud computing environments can benefit from moving towards Software-Defined Networking (SDN) technology. In SDN, the control logic is separated from individual forwarding devices, such as routers and switches, and implemented in a logically centralized controller. This allows the network control to be programmable and the underlying infrastructure to be abstracted for applications and network services. As a result, SDN allows network engineers and administrators to respond quickly to the changing business requirements by shaping traffic from the central controller without having to touch the physical switches. They use software to prioritize, redirect or block traffic either globally or in varying degrees down to individual packet levels.

There have been a number of works related to detection of network-based DDoS attacks in SDN. Phan *et al.* [18] introduce a hybrid approach based on combination of SVM and SOM [6] for flow classification in network traffic. Another work [22] suggests an attack detection system based on Bloom Filter and SDN to handle the link flooding attacks. In [11] a method based on SDN to detect DDoS attacks initiated by a larger number of bots for solving server attacks is proposed. The method uses the standard OpenFlow APIs designed for operation in general SDN environments. Other approaches related to detection of network-based DDoS attacks in SDN using machine learning techniques are described in [1,4,10]. To the best of our knowledge, there are only a few studies that try to detect application-based DDoS attacks in cloud environments with the help of SDN. Mohammadi *et al.* [16] present a software defined solution named Completely Automated DDoS Attack Mitigation Platform (CAAMP). When suspicious traffic is detected, CAAMP stores a copy of the original application on a private cloud and redirects suspicious traffic there. Thus, more time can be spent for processing suspicious traffic with no extra costs.

The aim of our research is to provide efficient and proactive solution for detecting application-layer DDoS attacks in cloud environment with the help of SDN. We propose a detection approach which is comprised of extracting normal user behavior patterns and detecting anomalies that significantly deviate from these patterns. This allows detection of attacks from legitimately connected network machines that are accomplished by using legitimate requests. Due to operating with information extracted from packet headers, the proposed scheme can be applied in secure protocols that encrypt the data of network connections without its decrypting. In order to evaluate our scheme, we implement a DDoS detection system prototype that employs the proposed algorithm. Simulation results show that intermediate application-layer DDoS attacks can be properly detected, while the number of false alarms remains very low. Finally, not only do we provide solution for detecting application-layer DDoS attacks in SDN-driven cloud environments, but also enhance the detection algorithm proposed in previous work [25]. These enhancements include:

1. Improved performance scores (FPR, TPR, accuracy).
2. Reduced number of parameters to effectively just one, which is the cluster number parameter in the first phase training when using k-Means. The second training phase is essentially parameterless.
3. Significant reducing the amount of storage needed by the detection algorithm. That is we only need to store centroids from the clustering phase and transition/marginal probability matrices from the second phase for each sequence length plus thresholds that are found automatically. Thus, the storage complexity is quadratic in the number of possible clusters $O(k^2)$, while it was at least $O(k^4)$.

The rest of the paper is organized as follows. Section 2 briefly describes the experiment setup. Section 3 summarizes main concepts and provides theoretical background of the proposed approach. Section 4 describes the algorithm proposed in the paper. Section 4.1 explains feature extraction process, while training and detection procedures are clarified in Sects. 4.2, 4.3 and 4.4. Meanwhile, Sect. 5 is devoted to the experimental results. It describes simulation environment, data set and results of the performance tests. Finally, Sect. 6 concludes the paper and outlines future work.

2 Problem Formulation

We consider a cloud environment in which cloud customers are allowed to create private virtual networks and connect them to the existing public networks with the help of virtual routers. In addition, every customer can spawn several virtual instances in own virtual networks. Each customer operates inside one of the projects created by a system administrator for a particular set of user accounts. We assume that neither user or administrator accounts have been compromised.

Further, we assume that networking inside the cloud is carried out with the help of SDN that includes an SDN controller and several SDN forwarding devices that are designed for working with virtual instances. SDN controller and switches communicate between each other inside the cloud management network and are not available directly from the data center VMs or external hosts. Scenarios in which either the controller or one of the switches is compromised are out of scope of this paper.

We consider a cloud customer that deploys several virtual web servers inside a virtual network providing access for other cloud customers as well as external hosts. Communication between the web servers and the users is carried out with encrypted traffic. Even though the web service provider relies on the data center security defenses, it cannot allow the cloud security engineers to decrypt the network traffic since it would violate regulations on privacy along with a high risk of conflict with the web service users. For this reason, detection of DDoS attacks is assumed to be carried out on network flow level.

In this study, we assume that network flows are captured on each SDN forwarding device and sent to the controller with the help of a NetFlow or sFlow agent. The controller investigates the received flow statistics and discovers behavior patterns of normal users. Once discovered, normal behavior patterns can be used to detect DDoS attacks against the web server applications and to block traffic from malicious cloud customers or external attackers in online mode.

3 Theoretical Background

3.1 *k*-Means-Based Clustering

k-Means [12,13] is one of the most popular algorithms for cluster analysis. It aims at partitioning data points into k clusters with the parameter k fixed a priory. Given a set of points $\chi = (\mathrm{x}_1, ..., \mathrm{x}_n)$, $\mathrm{x}_i \in \mathbb{R}^m$ the algorithm starts with initializing k centroids, one for each cluster, and assigning each data point x_i to the nearest centroid. Then iteratively the algorithm recalculates the centroids and re-assigns the data points to new clusters until convergence of the algorithm. Specifically, the algorithm aims at minimizing the sum of Euclidean distances between each data point and the mean value of the cluster this point belongs to, or to find

$$\arg\min_{\mathrm{C}} \sum_{i=1}^{k} \sum_{\mathrm{x} \in \mathrm{C}_i} \|\mathrm{x} - \mu_i\|^2,$$

where $\mathrm{C} = \{\mathrm{C}_1, ..., \mathrm{C}_k\}$ are data partitions and $\mu = (\mu_1, ..., \mu_k)$ are corresponding centroids.

3.2 CURE-Based Clustering

Despite traditional clustering methods have been widely used in data analysis, they have a number of drawbacks. For example, centroid-based methods, including k-Means, use only one point (centroid) to represent a cluster. If a cluster is large or has an arbitrary shape, the centroids of its subclusters can be distant from each other that could cause unnecessary splitting. On the opposite edge of the spectrum, all points-based methods such as k-NN or kernel, use all points for cluster representation and are sensitive to outliers and even slight changes in the position of data points. Both approaches fail to work well for defining non-spherical or arbitrarily shaped clusters [5].

Clustering Using REpresentatives (CURE) [5] is a hierarchical clustering algorithm which is a compromise between centroid-based and all point-based approaches and is suitable for large scale data sets. Compared to traditional methods, this approach is less sensitive to outliers and defines well even non-spherical clusters. First, initial clusters are created by hierarchical clustering of randomly picked sample points. Next, k scattered points describing a cluster shape and extent are picked, as disperse as possible. After shrinking towards the cluster centroid by a fixed fraction α these points become representatives of the cluster. When representative points are set up for each of the initial clusters the whole data set is rescanned and each point is assigned to the closest cluster. In traditional version of CURE the closest cluster for a point is defined as the closest one among all representative points of all the clusters. We modify the original procedure of cluster assignment as follows. After clusters are found, we take all representatives and centroids and continue using them as if they were an output of a centroid-based clustering algorithm, i.e., each centroid and/or representative is thought to be a center of a cluster. Such gradation of clusters allows better capturing complexity of user behavior types.

3.3 Probabilistic Transition-Based Approach for Detecting DDoS Attacks

Let $\mathbb{C} = \{c_i | i = 1, ..., K\}$ be a set of labels. Given a sequence of labels $c = (c_1, ..., c_N) \in \mathbb{C}^N$, let $P(c_i | c_{i-1}, l = N)$ denote conditional probability of observing label c_i after c_{i-1} in a sequence of length N. Marginal probability of observing label c_i at the beginning of the sequence is denoted as $P(c_i | l = N)$. We factorize joint probability distribution over sequences of length N as the following product:

$$P(c_1, ..., c_N | l = N) = P(c_1 | l = N) \times \prod_{i=2}^{N} P(c_i | c_{i-1}, l = N), \tag{1}$$

where l denotes length of the sequence. We estimate $P(c_i | c_{i-1}, l = N)$ as

$$P(c_i | c_{i-1}, l = N) \triangleq \frac{\mathrm{n}(c_{i-1}, c_i, N)}{\mathrm{n}(c_{i-1}, N)}, \tag{2}$$

where $\mathrm{n}(c_{i-1}, c_i, N)$ denotes count of observations of pairs (c_{i-1}, c_i) in all sequences of length N over all time windows and sessions, $\mathrm{n}(c_{i-1}, N)$ denotes count of observations of label c_{i-1} in all sequences of length N over all time windows and sessions. Moreover, $P(c_i | l = N)$ is estimated as

$$P(c_i | l = N) \triangleq \frac{\mathrm{n}(c_i, N)}{\sum_{j=1}^{K} \mathrm{n}(c_j, N)}. \tag{3}$$

Note, that in (3) we use the fact that the marginal probability of observing a label in a sequence should be equal to the marginal probability of observing the label at the beginning of a sequence since the windows are sliced arbitrarily.

During training phase we estimate conditional and marginal probabilities according to (2)–(3). Moreover, for every length of sequence N that is present in the training data we calculate minimal joint probabilities δ_N, which are further used as thresholds to examine new data for anomalies during test phase.

During test phase, we first calculate joint probability of a sequence of length N according to (1) and then compare it against a corresponding threshold value δ_N. If the sequence satisfies $P(c_1, ..., c_N | l = N) < \delta_N$ it is marked anomalous.

4 Algorithm

4.1 Feature Extraction

To detect outliers, we build a normal user behavior model. The features for building this model are extracted from a portion of network traffic at a very short time window that allows timely detection of attacks. The presented approach is based on the analysis of network traffic flows, namely, groups of IP packets with some common properties passing a monitoring point at a specified time interval. This time interval is defined to be equal to the time window. For analysis, we consider traffic flow extracted from the current time window. Furthermore, to reduce amount of data to be analyzed, we utilize aggregated traffic information by taking into account all packets of the flow transferred during previous time windows.

Next, we re-construct client to server conversations by combining the flow pairs such that the source socket of one flow equals to the destination socket of the other flow and vice versa. A conversation can be characterized by source IP, address, source port, destination IP address and destination port. For each such conversation, we extract the following information at every time interval:

1. Duration of the conversation.
2. Number of packets sent in 1 second.
3. Number of bytes sent in 1 second.
4. Average packet size.
5. Presence of packets with different TCP flags: URG, ACK, PSH, RST, SYN and FIN.

The set of features is defined by existing protocols for collecting IP traffic information such as NetFlow and sFlow. Since the values of the extracted feature vectors can have different scales, we standardize them using min-max normalization [6] by scaling to a range [0,1].

4.2 Training

We perform training using the standardized extracted features described in Sect. 4.1. First, we apply a clustering algorithm to divide the features into distinct groups representing specific classes of traffic in the network system. Thus, the algorithm discovers hidden patterns in the dataset. We assume that the traffic being clustered is mostly legitimate despite the fact it can be encrypted.

Therefore, we state that the obtained clusters describe behavior of normal users. Second, we group together conversations with the same source IP address, destination IP address and destination port extracted at a certain time interval. Such groups serve as an approximation of a user session and are analyzed separately, as other studies propose [3,23,24]. Next, we represent each session in every time window by a sequence of cluster labels obtained at the first step. Finally, from the obtained sequences we estimate conditional and marginal probabilities $P(c_i|c_{i-1}, l = N)$, $P(c_i|l = N)$ according to (2)–(3). For every sequence we calculate its probability using estimated parameters and model (1). In addition, we calculate thresholds δ_N by finding minimum among all sequence probabilities for a particular length of a sequence N.

4.3 Online Training Procedure

As behavioral patterns of users can change over time, we need to adapt our models in real-time. For adapting the clustering phase model we can use streaming k-Means algorithm. After clustering and classification have been done for a particular window t, one can update cluster centroids using the following formula:

$$\mu_i^{t+1} = \mu_i^t \cdot \delta + \sum_{\substack{\mathrm{x} \in \mathrm{C}_i^t, \\ \mathrm{x} \in \chi_{normal}}} \mathrm{x} \cdot (1 - \delta),$$

where μ_i^t is centroid of the cluster i at the time window t, C_i^t is the set of data points assigned to the cluster i, χ_{normal} is a set of data points classified as normal, and $\delta \in [0, 1]$ is a constant reflecting how fast the model has changed when a new observation emerged, i.e. for bigger δ the model changes slower. For CURE the same formula can be used, but representatives are updated instead of the cluster centroids.

To update transition probabilities from the probabilistic model dynamically, we apply the following updates that are performed for each pair of labels (c_{i-1}, c_i) from the label sequences that were classified as normal:

$$P(c_i|C_{i-1} = c_{i-1}, l = N) \leftarrow P(c_i|C_{i-1} = c_{i-1}, l = N) + \epsilon,$$
$$P(c_i|C_{i-1} \neq c_{i-1}, l = N) \leftarrow P(c_i|C_{i-1} \neq c_{i-1}, l = N) - \epsilon/(K - 1),$$

where C_{i-1} is a random variable that denotes cluster label at position $i - 1$ and ϵ represents the velocity of change of a conditional probability once a new evidence has been observed. Thus, ϵ affects how fast model is changed with respect to new data. These updates guarantee that the conditional probability remains properly normalized by adding a probability mass to the parameter that accounts for the new data and removing the same amount of probability mass from the parameters that do not correspond to the new data. Moreover, these updates implement a forgetting mechanism as the old evidence gets less and less influence on the model with time.

In order to keep thresholds δ_N up to date we propose to store top N_δ data sequences in a heap data structure with keys equal to probabilities of the data sequences. We need to keep a separate heap for label sequences of each length. Every time the model is updated the top element with the lowest probability (equal to the current threshold δ_N) is popped out and pushed in the heap again with a new recomputed probability key. Moreover, the threshold δ_N is assigned the new value. This way threshold can either become bigger or smaller. Threshold value is also updated once a new normal data sequence gets smaller probability under the current model.

4.4 Detection

For detecting anomalies we use a model of normal user behavior obtained during training phase. First, we assign each session with a sequence of cluster numbers using clustering model from the training phase. Then, similarly to the training phase, we calculate probability of every sequence using estimated probability parameters and the model (1). The obtained probability values are compared against thresholds δ_N to decide whether the sequence is anomalous or not. If probability of a sequence is less than a threshold probability then it is marked as anomaly.

5 Algorithm Performance

5.1 Simulation Environment and Data Set

We test the attack detection algorithm proposed in this study in a virtual network environment that includes a small botnet, command and control center (C2) and a target web bank server (see Fig. 1). The target server is running in the Openstack [7] cloud environment where networks are carried out by an Opendaylight [9] integrated SDN controller and several Open vSwitches [17]. Bots and C2 are located outside the cloud. Each bot is a VM with running a special program implemented in Java, it receives commands from C2 and generates some traffic to the server. It is worth noting that all the traffic is transfered by using encrypted SSL/TLS protocol. All network flows are captured on SDN switches and sent to the controller with the help of NetFlow agents.

In order to generate a normal bank user traffic, we specify several scenarios that each bot follows when using the bank site. Each such scenario consists of several actions following each other. The list of the actions consists of logging in to the system by using the corresponding user account, checking the account balance, transferring some money to another account, checking the result of the transaction, logging out of the system, and some other actions. Each action corresponds to requesting a certain page of the bank service with all of its embedded content. Pauses between two adjacent actions are selected in a way similar to a human user behavior. For example, checking the account balance usually takes only a couple of seconds, whereas filling in information to transfer money to another account may take much longer time.

In addition to the normal traffic, we perform an intermediate DDoS attack during which several bots-attackers try to mimic the browsing behavior of regular

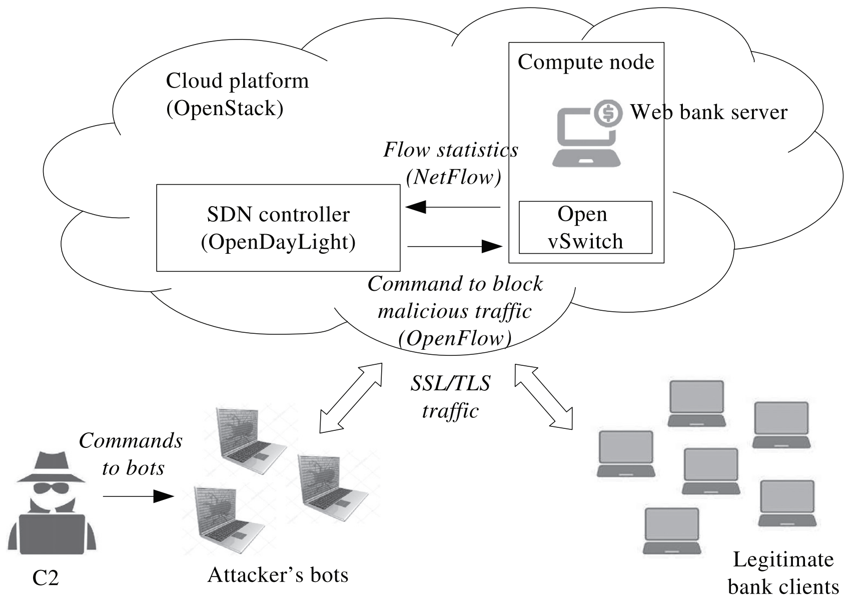

Cloud platform (OpenStack)
Compute node
Web bank server
Flow statistics (NetFlow)
SDN controller (OpenDayLight)
Open vSwitch
Command to block malicious traffic (OpenFlow)
SSL/TLS traffic
Commands to bots
C2
Attacker's bots
Legitimate bank clients

Fig. 1. Virtual network simulation environment.

users by requesting sequences of web pages with all embedded content from the service. However, unlike the normal user behavior, these sequences are not related to each other by any logic but generated randomly. We consider the case when the attacker sends traffic with about the same rate as normal users, and each attacker's connection individually looks like normal. More advanced attack scenarios are left for future works.

5.2 Results

We evaluate the proposed approach on the test set described in Sect. 5.1. We propose two methods for detecting intermediate DDoS attacks which both consist of two phases. The first method (k-Means+Prob) uses k-Means clustering in the first phase and probabilistic transition-based approach (Prob) in the second phase. The second method (CURE+Prob) applies CURE clustering in the first phase and Prob in the second phase. The algorithms have been evaluated using the detection accuracy, true positive rate (TPR) and false positive rate (FPR) performance metrics [6].

In our experiments, the time window size is set to 5 seconds, due to the nature of the data. Moreover, we are only interested in results when FPR is below 1% as the high number of false alarms is one of the most important known drawbacks of anomaly-based detection systems.

Table 1 displays accuracy of detecting intermediate DDoS attacks for the proposed detection schemes. For comparison, we also include to Table 1 performance

Table 1. Accuracy of detecting intermediate DDoS attacks

Algorithm	TPR (%)	FPR (%)	Accuracy (%)
k-Means+Prob	98.66	0	99.58
CURE+Prob	95.65	0	86.16
2-gram k-Means	95.08	0.24	86.61
3-gram k-Means	91.21	0	73.66

results of the k-Means-based data stream clustering approach proposed in [25]. The parameters of the methods are selected to maximize the detection accuracy on validation set. The best result is shown by the (k-Means+Prob) approach which outperforms other methods by 13% in terms of accuracy. Still, other methods perform relatively well reaching accuracy of 86% with FPR equaling to or near zero.

To visualize the results, we plot ROC curves in Fig. 2(a). ROC curves corresponding to the (k-Means+Prob) and (CURE+Prob) methods proposed in this paper are displayed by dashed and plain lines, correspondingly. Furthermore, by dash-dot and dotted lines we plot ROC curves for the k-Means-based data stream clustering approach proposed in [25] for 2-gram and 3-gram models, respectively. From the ROC curves one can see that (k-Means+Prob) is the only among the presented algorithms that reaches TPR of 100%. Other methods demonstrate similar performance reaching the highest TPR of around 98% at FPR near 1.5%.

In addition, for the best performing algorithm (k-Means+Prob) we plot how performance scores depend on number of clusters, which is the only parameter of this method. Figure 2(b) shows that the algorithm performs relatively well for all parameter values reaching the maximum in accuracy and TPR when the number of clusters is equal to 12. FPR, which is plotted in (100%−FPR) scale for better visual representation, remains below 1% for all parameter values.

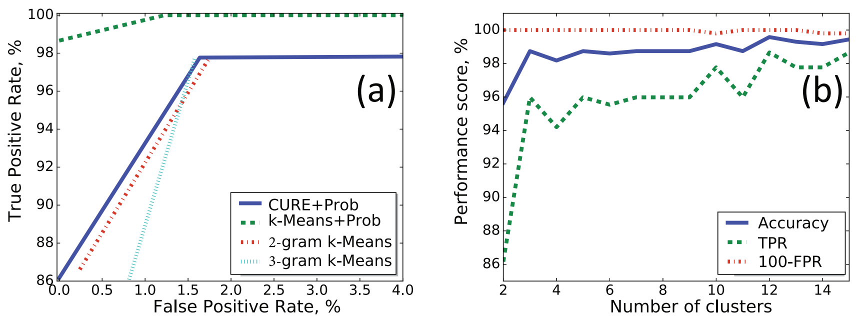


Fig. 2. (a) - ROC curves for detection of intermediate DDoS attacks, (b) - dependence of performance scores from number of clusters for the (k-Means+Prob) algorithm.

6 Conclusions and Future Work

In this work, we proposed probabilistic transition-based approach for detecting intermediate application-layer DDoS attacks in cloud environment with the use of SDN. Operating with information extracted from the packet headers makes this approach suitable for detecting DDoS attacks from encrypted traffic. We tested the proposed algorithms against other methods used for detecting application-layer DDoS attacks in encrypted networks proposed earlier in [25]. Both presented algorithms demonstrated good performance results. Moreover, (k-Means+Prob) significantly outperforms other evaluated algorithms under the condition of FPR $< 1\%$.

In the future, we plan to improve the algorithm in terms of the detection accuracy and test it with a bigger dataset. In addition, more focus will be on the simulation and detection of more advanced DDoS attacks.

Acknowledgment. This research was supported by the Nokia Foundation Scholarship funded by Nokia, Finland.

References

1. Chen, P.J., Chen, Y.W.: Implementation of SDN based network intrusion detection and prevention system. In: 2015 International Carnahan Conference on Security Technology (ICCST) (2015). https://doi.org/10.1109/CCST.2015.7389672
2. Chen, R., Wei, J.Y., Yu, H.F.: An improved grey self-organizing map based dos detection. In: IEEE Conference on Cybernetics and Intelligent Systems, pp. 497–502 (2008). https://doi.org/10.1109/ICCIS.2008.4670765
3. Chwalinski, P., Belavkin, R., Cheng, X.: Detection of application layer DDoS attacks with clustering and Bayes factors. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 156–161 (2013). https://doi.org/10.1109/SMC.2013.34
4. Dotcenko, S., Vladyko, A., Letenko, I.: A fuzzy logic-based information security management for software-defined networks. In: 16th ICACT, pp. 167–171 (2014). https://doi.org/10.1109/ICACT.2014.6778942
5. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. Inf. Syst. **26**(1), 35–58 (2001). doi:10.1016/S0306-4379(01)00008-4
6. Hastie, T.J., Tibshirani, R.J., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, New York (2009). doi:10.1007/978-0-387-84858-7
7. Jackson, K.: OpenStack Cloud Computing Cookbook. Packt Publishing, Birmingham (2012)
8. Ke-Xin, Y., Jian-qi, Z.: A novel dos detection mechanism. In: International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), pp. 296–298 (2011). https://doi.org/10.1109/MEC.2011.6025459
9. Knorr, E.: Opendaylight: A big step toward the software-defined data center. InfoWorld (2013)
10. Le, A., Dinh, P., Le, H., Tran, N.C.: Flexible network-based intrusion detection and prevention system on software-defined networks. In: 2015 ACOMP, pp. 106–111 (2015). https://doi.org/10.1109/ACOMP.2015.19

11. Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S.: A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: 2014 6th International Conference on Ubiquitous and Future Networks (ICUFN), pp. 63–68 (2014). https://doi.org/10.1109/ICUFN.2014.6876752
12. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theor. **28**(2), 129–137 (2006). https://doi.org/10.1109/TIT.1982.1056489
13. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
14. Mills, K., Yuan, J.: Monitoring the macroscopic effect of DDoS flooding attacks. IEEE Trans. Dependable Secure Comput. **2**, 324–335 (2005). https://doi.org/10.1109/TDSC.2005.50
15. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. **34**(2), 39–53 (2004). http://doi.acm.org/10.1145/997150.997156
16. Mohammadi, N.B., Barna, C., Shtern, M., Khazaei, H., Litoiu, M.: CAAMP: completely automated DDoS attack mitigation platform in hybrid clouds. In: 12th International CNSM, pp. 136–143 (2016). https://doi.org/10.1109/CNSM.2016.7818409
17. Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., Casado, M.: The design and implementation of open vswitch. In: 12th USENIX Conference on Networked Systems Design and Implementation (NSDI), pp. 117–130 (2015)
18. Phan, T.V., Bao, N.K., Park, M.: A novel hybrid flow-based handler with DDoS attacks in software-defined networking. In: 2016 IEEE UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld (2016). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069
19. Radware: 2015–2016 global application & network security report. https://www.radware.com/newsevents/pressreleases/radwares-2015-2016-global-applications-and-network-security-report/
20. Somani, G., Gaur, M.S., Sanghi, D., Conti, M., Buyya, R.: DDoS attacks in cloud computing: issues, taxonomy, and future directions. ACM Comput. Surv. **1**(1), 1–44 (2015)
21. Stevanovic, D., Vlajic, N.: Next generation application-layer DDoS defences: applying the concepts of outlier detection in data streams with concept drift. In: 13th ICMLA, pp. 456–462 (2014). https://doi.org/10.1109/ICMLA.2014.80
22. Xiao, P., Li, Z., Qi, H., Qu, W., Yu, H.: An efficient DDoS detection with bloom filter in SDN. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 1–6 (2016). https://doi.org/10.1109/TrustCom.2016.0038
23. Xu, C., Zhao, G., Xie, G., Yu, S.: Detection on application layer DDoS using random walk model. In: IEEE International Conference on Communications (ICC), pp. 707–712 (2014). https://doi.org/10.1109/ICC.2014.6883402
24. Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J.: Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. In: 23rd ICT, pp. 1–6 (2016). https://doi.org/10.1109/ICT.2016.7500408
25. Zolotukhin, M., Kokkonen, T., Hämäläinen, T., Siltanen, J.: On application-layer DDoS attack detection in high-speed encrypted networks. Int. J. Digital Content Tech. Appl. **10**(5), 14–33 (2016)