UNIVERSITÀ DELLA CALABRIA

**UNIVERSITA' DELLA CALABRIA**

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

**Dottorato di Ricerca in**

Information and Communication Engineering for Pervasive Intelligent Environments

**CICLO**

XXIX

**TITOLO TESI**

**"Data Mining Techniques for Large and Complex Data"**

**Settore Scientifico Disciplinare ING-INF/06**

**Coordinatore:** Ch.mo Prof. FELICE CRUPI

Firma Felice Crupi

**Supervisore/Tutor:** Ch.mo Prof. FABRIZIO ANGIULLI

Firma F. Angiulli

**Dottorando:** Dott./ssa ESTELA NARVAEZ

Firma Estela Narvaez

## Acknowledgements

*First and foremost, I want to thank to our holy God for giving me the strength I needed to reach this goal. It has been a long and so difficult road but finally, I got it. This would not have been possible without the assistance of Professor Fabrizio Angiulli. Professor, I thank you from the bottom of my heart for all the valuable input, guidance, and helpful comments regarding to this thesis work and also throughout these three years. My gratitude to my family for their priceless support and for the countless sacrifices they have made to enable me to accomplish with this phase of my professional life. Thanks for their endless love and motivation, for being with me in my ups and downs, for encouraging me to pursue my dreams. Finally, I would like to thank to all my friends who were my family during this time in Italy, for their backing during hard times, and for making this an unforgettable experience.*

# Contents

# List of Figures

# List of Tables

# 1

# Abstract

Questo lavoro di tesi ha riguardato tecniche di data mining per dati complessi e di grandi dimensioni. Sono state considerate due famiglie di algoritmi di data mining: la classificazione e l'analisi di anomalie.

La prima parte di questa tesi si concentrata sulla progettazione di strategie per ridurre la dimensione del sottogruppo estratto tramite tecniche di condensazione e sulla loro successiva sperimentazione.

Il risultato della ricerca stato lo sviluppo di varie strategie di selezione di sottoinsiemi, progettate per determinare, durante la fase di training, il sottoinsieme pi promettente sulla base di diversi metodi di stima della precisione in fase di test. Tra questi, la strategia PACOPT basata sul Pessimistic Error Estimate (PEE) per stimare il grado di generalizzazione come trade-off tra laccuratezza del training set e la complessit del modello. La fase sperimentale ha avuto come riferimento la tecnica di condensazione FCNN. Tra i metodi di condensazione basati sulla regola di decisione nearest neighbor (NN rule), FCNN (acronimo di Fast Condensed NN ) una delle tecniche pi vantaggiose, in particolare per quanto riguarda le prestazioni relative al tempo. Si mostrato che le strategie di selezione progettate consentono di preservare la precisione di ciascun sottogruppo consistente. Si dimostrato, inoltre, che le strategie di selezione proposte garantiscono una riduzione significativa delle dimensioni del modello. Infine, alcune tra le principali tecniche di riduzione del training-set per la NN rule, che rappresentano lo stato dellarte in termini di prestazioni, sono state confrontate con le strategie qui proposte.

La seconda parte della tesi diretta alla progettazione di strumenti di analisi per i dati strutturati sotto forma di rete.

Lanomaly detection un'area che ha ricevuto molta attenzione negli ultimi anni. Ha una vasta gamma di applicazioni, tra cui la rilevazione di frodi e il rilevamento delle intrusioni in rete. Le tecniche focalizzate sul rilevamento di anomalie nei grafi statici presuppongono che le reti non cambino e siano in grado di rappresentare solo una singola istantanea dei dati. Poich le reti del mondo reale sono in continua evoluzione, stata spostata lattenzione sui grafi dinamici, che si evolvono nel tempo. E stata presentata una tecnica per

lanomaly detection di nodi in reti in cui gli archi sono etichettati con listante di tempo di creazione. La tecnica mira a individuare anomalie prendendo contemporaneamente in considerazione le informazioni relative alla struttura della rete e all'ordine in cui sono state stabilite le connessioni. Lobiettivo primario quello di analizzare ogni singolo nodo, considerando contemporaneamente la sua dimensione temporale. Questultima informazione stata ottenuta tramite i timestamps associati agli archi. E possibile indurre un insieme di strutture temporali controllando determinate condizioni sull'ordine di apparizione di ciascun arco, poich ci denota diversi tipi di comportamento da parte dellutente. La distribuzione di queste strutture calcolata per ciascun nodo e usata per individuare le anomalie.

# 2

# Introduction

Data mining extracts useful information from large amounts of data and it is one of the fast growing fields in the information technology. There is a huge amount of information locked up in databases which have not been discovered or analyzed, to the date. This thesis aims to provide novel techniques for large and complex data, focusing on issues relating to Classification and Anomaly Detection.

One of the most important descriptive data mining tasks is classification, which can be used to find the differentiating features among classes and to exploit them to classify unknown instances. Given a training set, that is a set of examples for which class labels are known and given a learning algorithm, examples are used to build a classification model that is applied to the test set, or the set of observations for which class labels are unknown. The nearest neighbor rules are based on learning by analogy, that is by comparing a given test tuple with the training tuples that are similar to it. Within this domain, the main purpose of this thesis was to study algorithms able to improve the classifier accuracy and to prevent the induction of overly complex models in the context of nearest neighbor condensing classification techniques. Specifically, it was investigated the application of the pessimistic error estimate principle to minimize the expected error rate of the classifier.

The second part of the thesis is directed towards the design of analysis tools for network structured data, such as the discovery of anomalies within a network. Social networks are formally defined as a set of nodes that are tied by one or more types of relations. Social networking sites are examples of widely popular networks used to find and organize contacts. Identifying anomalies has become a challenge and many studies have been conducted to face this matter. This thesis intends to give a contribution in this setting by describing techniques for node anomaly detection in networks that take into account both structural and temporal information.

The rest of thesis is organized in two main parts, corresponding to the two above main data mining tasks faced in the context of large and complex data.

# Part I

# Classification Techniques for Large Datasets

# 3

# Pruning Nearest Neighbor Competence Preservation Learners

The nearest neighbor decision rule [22] (NN rule for short) assigns to an unlabelled sample point the classification of the nearest of a set of previously classified points. A strong point of the nearest neighbor rule is that, for all distributions, its probability of error is bounded above by twice the Bayes probability of error [22, 86, 25]. That is, it may be said that half the classification information in an infinite size sample set is contained in the nearest neighbor.

Naive implementation of the NN rule requires to store all the previously classified data points, and then to compare each sample point to be classified to each stored point.

In order to reduce both space and time requirements the concept of *training set consistent subset*, that is a subset of the original training set that correctly classifies all the training samples, was introduced by Hart [47] together with an algorithm, called the CNN rule (for Condensed NN rule), to determine a consistent subset of the original sample set. Since then different techniques have been introduced [101, 56, 24, 8] with the same goal, referred to as training set reduction, training set condensation, reference set thinning, or prototype selection algorithms.

A *training set consistent subset* is a subset of the training set that classifies the remaining data correctly through the NN rule.

Using a training set consistent subset, instead of the entire training set, to implement the NN rule, has the additional advantage that it may guarantee better classification accuracy. Indeed, [56] showed that the VC dimension of an NN classifier is given by the number of reference points in the training set.

Thus, in order to achieve a classification rule with controlled generalization, it is better to replace the training set with a small consistent subset.

Motivated by approaches used in the context of other classification algorithms in order to improve generalization and to prevent induction of overly complex models, such as in the case of decision trees, we investigate the application of the Pessimistic Error Estimate (PEE) principle [91] in the context of the nearest neighbor rule.

The rest of this first part is organized as follows. Chapter 4 presents an overview of the relevant background and gives a brief account of the prominent works conducted in these areas in the last few years. Contributions are detailed in subsequent Chapter 5.

# 4

# Scenario and Related Work

Data Mining is a discipline used in various domains to give meaning to the available data. In this chapter, we will review the necessary background as well as previous work within the main areas of this thesis. Focusing first on the classification. Next, we present an overview of techniques for Nearest Neighbor classification focusing on $k$-NN classifiers and Nearest Neighbor Condensation Techniques, in addition to methods for evaluating and comparing the performance of the classification techniques, and to techniques for reducing storage requirements and improving classifier accuracy.

## 4.1 Classification

Classification is a data mining technique, whose goal is to assign objects to one of several predefined categories. Classification is a pervasive problem that encompasses many diverse applications such as, Business, Games, Science and engineering, Medical, Spatial to cite a few [103].

Classification is defined [46] as the process of finding a model (or function) that describes and distinguishes data classes and concepts. This model can also be used to predict the class label of objects whose class membership is unknown. Classification techniques employ a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data.

**Definition 4.1.** *Classification is the task of learning a* ***target function*** $f$ *that maps each attribute set* $\boldsymbol{x}$ *to one of the predefined class labels* $y$*. [91]*

The target function is also called as a classification model. This model can serve as an explanatory tool to distinguish between objects of different classes. Classification techniques are most fitted for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories because they do not consider the implicit order among the categories. Other

forms of relationships, such as the subclass-superclass relationships among categories are also ignored.

During the years a lot of different classification algorithms have been proposed, including decision tree classifiers, K-nearest neighbor classifiers, rule-based classifiers, Neural Networks, Support Vector Machines and Naive Bayesian classifier [46]. Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.

Classification methods have been proposed by researchers in machine learning, pattern recognition, data mining and statistics.

The rest of the section introduces the problem of classification, some major classification methods, including Decision Trees induction $k$-Nearest Neighbor classifiers, the main measures of accuracy as well as techniques for obtaining reliable accuracy estimates and for increasing classifier accuracy.

### 4.1.1 Decision Trees

Decision trees are one of the most common classification techniques used in the practice. A decision tree is a hierarchical structure consisting of nodes and directed edges.

Decision trees build classification or regression models in the form of a tree structure. As a matter of fact, a decision tree is a flowchart-like tree structure. The tree has three types of nodes: Root node that has no incoming edges and one or more outgoing edges. Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges. Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges [91]. Each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf (or terminal node) holds a class label. A decision tree can be adapted so as to predict continuous (ordered) values, rather than class labels [46].

Decision trees that are too large are susceptible to a phenomenon known as overfitting the data. To overcome this problem, pruned trees are used which trees tend to be smaller and less complex. One of the strong points of these algorithms is that they are usually faster and better at correctly classifying independent test data.

### 4.1.2 Nearest Neighbor Classification

The nearest neighbor (NN) technique is one of the oldest classification methods known and one of the most useful algorithms in data mining in spite of being very simple and effective for performing recognition tasks.

The nearest neighbor decision rule [22] assigns to an unclassified sample point the classification of the nearest of a set of previously classified points. The nearest neighbor classification method belongs to the category of instance-based learning methods, since the entire training set is stored and each new instance is compared with existing ones using a distance metric, and the closest existing instance is used to assign the class to the new one. These classifiers therefore can suffer from poor accuracy when given noisy or irrelevant attributes. Sometimes more than one nearest neighbor is used, and the majority class of the closest $k$ neighbors is assigned to the new instance. This is termed the $k$-nearest neighbor method [103].

The classification error of the nearest neighbor rule is bounded above by twice the optimal Bayes error under certain reasonable assumptions. Furthermore, the error of the general $k$-NN method asymptotically approaches that of the Bayes error and can be used to approximate it [22].

### 4.1.3 Evaluation Methods

There are many evaluation methods that can be applied to improve the performance of classifiers in data mining. We review the principal methods used to evaluate the performance of a classifier.

#### Holdout method

In this method [91] the original data set is partitioned into two parts, the training set and test set. The classification model is induced from the training set and evaluated on the test set. The proportion of data reserved for training and for testing is typically at the choice of the analysts. In this method, the dataset is usually randomly divided, two-thirds for training and one-third of the data for testing. The holdout method can be repeated various times to improve the estimation of classifier's accuracy.

#### Cross Validation

Cross Validation (CV) [87] is one of the most widely used methods to measure model prediction performance. In this approach, each record is used the same number of times for training and exactly once for testing. The basic idea of this method [91] is to split the data on a fixed number of folds, or partitions.

In k-Fold Cross Validation [46] the data set is randomly split into $k$ subsets or folds of approximately equal size. At each run, one of the partitions is chosen for testing, while the rest of them are used for training. This procedure is repeated $k$ times such that each fold is used for testing exactly once. Finally, the total error is found by summing up the errors for all $k$ runs in each fold.

In stratified cross-validation, the folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data. Generally, stratified 10-fold cross validation is recommended for estimating accuracy due to its relatively low bias and variance.

**Leave-One-Out Cross Validation**

The leave-one-out cross validation is simply $n$-fold cross-validation [103], where $n$ is the number of instances in the data set. Each instance in turn is left out, and the learning scheme is trained on all the remaining instances. This technique can usually be applied only on small datasets due that it is computationally costly to repeat the procedure $n$ times. Nevertheless, leave-one-out seems to offer the opportunity to squeeze the maximum out of a small dataset and getting as accurate an estimate as possible.

## 4.2 Overfitting and Model Complexity

Errors committed by classification models [103] are generally divided into two types: Training error and Generalization error. The training error also known as resubstitution error, is the number of misclassification errors committed on training records, whereas the generalization error is the expected error of the model on previously unseen records. A good model must have low training error as well as low generalization error. This is important because a model that fits the training data too well can have a poorer generalization error than a model with a higher training error, this phenomenon is known as model overfitting. The model overfitting problem has been investigated by several authors including [81, 26, 53].

The chance for model overfitting increases as the model becomes more complex. For this reason, might be preferred simpler models, a strategy that agrees with a well-known principle known as Occam's razor:

**Definition 4.2.** ***Occam's Razor:*** *Given two models with the same generalization errors, the simpler model is preferred over the more complex model [91].*

Thus, Overfitting happens when a model is more flexible than it needs to be and incorporates noise in the training data to the extent that it negatively impacts the performance on the model on new data [48]. There are several possible causes why overfitting happens: the presence of noise, a model too complex, a small training set, a very rich hypothesis space, a domain with many features [91]. In order to prevent overfitting, it is necessary to use additional techniques as cross-validation [50, 52], regularization [10], early stopping [78], pruning [55], and Bayesian approach [31].

A way to help learning algorithms to select the most appropriate model is to be able to estimate the generalization error. The right complexity is that of a model that produces the lowest generalization error. The problem is that the learning algorithm has access only to the training set during model building. It has no knowledge of the test set, and thus, does not know how well the tree will perform on records it has never seen. The best is to estimate the generalization error of the induced tree. There are several methods for

doing the estimation that are *Resubstitution Estimate, Estimating Statistical Bounds, and Validation Set. Resubstitution Estimate*, assumes that the training set is a good representation of the overall data, so they use the training error to provide an optimistic estimate that this value can be used to characterize the generalization error. The training error is usually a poor estimate of generalization error.

*Estimating Statistical Bounds*, the generalization error can also be estimated as a statistical correction to the training error. This statistical correction is computed as an upper bound to the training error, taking into account the number of training records that reach a particular leaf node.

*Validation Set*, this method divides the original training data into smaller subsets. One of the subsets is used for training, while the other, is used for estimating the generalization error [91].

## 4.3 Reducing Complexity of Decision Trees

A problem common when a decision tree is built is that many of the branches will reflect anomalies in the training data due to noise or outliers. As the number of nodes in the decision tree increases, the tree will have fewer training errors. Up to a certain size also the test error will decrease. However, once the tree becomes too large, its test error rate begins to increase even though its training error rate continues to decrease. Such a situation is known as model overfitting [91].

Various methods can be employed to combat this problem.

Two well-known techniques for incorporating model complexity into the evaluation of classification models are PEE and MDL, which are described next. In the context of decision trees.

### 4.3.1 Pessimistic Error Estimate.

The first approach explicitly computes generalization error as the sum of training error and a penalty term for model complexity. The resulting generalization error can be considered its pessimistic error estimate. For instance, Let $n(t)$ be the number of training records classified by node $t$ and $e(t)$ be the number of misclassified records. The pessimistic error estimate of a decision tree $T$, $e_g(T)$, can be computed as follows:

$$e_g(T) = \frac{\sum_{i=1}^{k}[e(t_i) + \Omega(t_i)]}{\sum_{i=1}^{k} n(t_i)} = \frac{e(T) + \Omega(t)}{N(t)},$$

where $k$ is the number of leaf nodes, $e(T)$ is the overall training error of the decision tree, $N_t$ is the number of training records, and $\Omega(t_i)$ is the penalty term associated with each node $t_i$. For example, consider the binary decision

trees. If the penalty term is equal to 0.5, $e(T) = 6$, $\Omega(t) = 4$, and $N(t) = 24$ then the pessimistic error estimate is

$$e_g(T) = \frac{6+4*0.5}{24} = \frac{8}{24} = 0.3333$$

### 4.3.2 Minimum Description Length Principle.

MDL principle is a general method for inductive inference, based on the idea that the more we are able to compress a set of data, the more regularities we have found in it [43]. Is based on an information-theoretic approach, when two models fit the data equally well, MDL will choose the one that is the simplest in the sense that it allows for a shorter description of the data [91]. The MDL principle, provides a unified approach to statistical modeling, and it allows the estimation of parameters along with their number without separate hypothesis testing. It involves adding to the error function an extra term which is designed to penalize mappings which are not smooth [79, 10]. MDL principle use encoding techniques to define the best decision tree as the one that requires the fewest number of bits to both 1) encode the tree and 2) encode the exceptions to the tree. Its main idea is that the simplest of solutions is preferred [46].

### 4.3.3 Pruning Decision Trees

The most common approach to construct decision tree classifiers is to grow a full tree and prune it back. Pruning is desirable because the tree that is grown may overfit the data by inferring more structure than is justified by the training set [11]. There are two common approaches to tree pruning: *Pre-pruning* and *Post-pruning*.

The first approach stops growing the tree earlier, before it perfectly classifies the training set. Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.

The second approach, post-pruning, removes subtrees from a fully grown tree. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced [46].

Several pruning methods have been introduced in the literature, including reduced error pruning [75], pessimistic error pruning [73], minimum error pruning [69], critical value pruning [65], cost-complexity pruning [12], penalty pruning [60], Error-Based Pruning [74]. Esposito *et al.* [30] make a comparative study of pruning methods cited above with the aim of understanding their theoretical foundations, their computational complexity, and the strengths and weaknesses of their formulation.

Alternatively, pre-pruning and post-pruning may be interleaved for a combined approach. Post-pruning requires more computation than pre-pruning, yet generally leads to a more reliable tree. The advantage of pre-pruning is that it avoids generating overly complex subtrees that overfit the training data. Post-pruning tends to give better results than pre-pruning since it makes pruning decisions based on a fully grown tree, unlike pre-pruning, which can suffer from premature termination of the tree-growing process [91].

## 4.4 Reducing Complexity of Nearest Neighbor Classifiers

In order to reducing complexity of Nearest Neighbor classifiers, three categories of techniques exist for the selective storage of instances, i.e. *competence enhancement*, *competence preservation*, and *hybrid approaches*. Competence enhancement methods remove noisy or corrupted instances such that classification accuracy is improved. Competence preservation remove superfluous instances such that performance is improved while classification accuracy of the training set is preserved. Finally hybrid approaches perform both competence enhancement and competence preservation [13].

One of the first competence enhancement techniques is Edited Nearest Neighbor (ENN) algorithm [102], which removes instances considered to be noise from the training set; an instance is considered as noise if it does not agree with its $k$ nearest neighbors. Tomek [93] introduced Repeated-ENN (RENN) is extension of the ENN algorithm and All k-NN (ANN) algorithms. Both make multiple passes over the training set, the former repeating the ENN algorithm until no further eliminations can be made from the training set and the latter using incrementing values of $k$.

The seminal work in competence preservation is the Condensed Nearest Neighbor (CNN) [47]. Gates [40] devised the Reduced Nearest Neighbour (RNN) rule, which extends the idea of the CNN. The Selective Nearest Neighbor Rule (SNN) devised by Ritter *et al.* [80] improves on the CNN and RNN.

Aha *et al.* [2], introduced a series of instance-based learning algorithms to reduce storage requirements and tolerate noisy instances. Other researchers have provided variations on the IBn algorithms [105, 14]. Wilson and Martinez [100] introduced three new instance reduction techniques which are intuitive and provide good storage reduction called RT1, RT2 and RT3. RT3, have shown higher generalization accuracy and lower storage requirements.

### 4.4.1 Nearest Neighbor Condensation Techniques

#### Condensed Nearest Neighbor Rule

The Condensed Nearest Neighbor rule (CNN rule) was proposed by Hart [47], in order to determine a consistent subset of the original sample set, with the goal of reducing the storage requirements involved in the NN rule. The aim

of the CNN algorithm is to minimize the number of sample stored, keeping only a subset of training data for classification. The basic idea is to remove the data points which do not add additional information and show similarity with other training data set. A minimal consistent subset is a subset of the training set having the property of allowing correct classification of the training set objects through the use of the NN rule.

The algorithm uses two bins, called $S$ and $T$ and proceed as follows. The first sample pattern is copied from $S$ to $T$. $T$ is used as the training set to classify each pattern of $S$, starting with the first. This is done until one of the following two cases arises: If all samples are classified correctly, the process finishes; if a point is incorrectly classified, it is added to $T$. The algorithm terminates when no misclassified point is added during a complete pass of $S$. In general the CNN rule adds prototypes until a stop condition is met.

The CNN rule may select points far from the decision boundary. Besides, the CNN rule is order dependent, that is it has the undesirable property that the consistent subset depends on the order in which the data is processed.

#### Reduced Nearest Neighbor Rule

Gates [40] introduced the Reduced Nearest Neighbor Rule (RNN) as an improvement over CNN. The RNN algorithm starts with $S = T$ where $T$ is the whole training set and removes each pattern from $S$ if whose elimination does not affect the classification of the training data set $T$. This algorithm is able to eliminate noisy instances and internal instances while retaining border points. From the perspective of temporal cost, it is more expensive than the CNN rule. It will always produce a subset of the results of CNN algorithm.

#### Selective Nearest Neighbor Decision Rule

Ritter *et al.* [80] propose Selective Nearest Neighbor Rule (SNN). This algorithm produces a selective subset of the original data so that 1) the subset is consistent, 2) the distance between any sample and its nearest selective neighbor is less than the distance from the sample to any sample of the other class, and 3) the subset is the smallest possible.

#### Modified Condensed Nearest-Neighbor Rule

The Modified Condensed Nearest-Neighbor Rule [24] (MCNN rule) algorithm builds a training set in an incremental manner. MCNN rule starts with a basic set comprising one pattern from each class. This algorithm is based on the misclassified samples, a representative prototype of each class is determined and added to the set of basic prototypes to classify these patterns correctly. The process is repeated until there are no misclassified points in training set. The algorithm is fast and order independent.

The MCNN algorithm may work well in the case of patterns with a Gaussian distribution with an equal and diagonal covariance matrix. In every iteration of the MCNN rule, a set of representative samples is found for a group of patterns. Thus, the algorithm might require a lot of iterations to converge, due to the large number of features.

### Decremental Reduction Optimization Procedure Algorithms

In [101] suggests a series of six algorithms for sets reduction based on the kNN algorithm called DROP1, DROP2, DROP3, DROP4, DROP5 and DEL.

DROP1. This algorithm is identical to RNN rule, with the exception that the accuracy is analyzed on $S$ instead of $T$. It uses the following basic rule to decide if it is safe to eliminate an instance from the instance set S (where S = T originally): eliminate $P$ only if at least some of its associates in $S$ can be classified correctly without $P$.

DROP2. This algorithm removes $P$ if at least as many of its associates in $T$ would be classified correctly without $P$. Using this modification, each instance $P$ in the original training set $T$ continues to maintain a list of its $k+1$ nearest neighbors in $S$, even after $P$ is removed from $S$. This means that instances in $S$ have associates that are both in and out of $S$, while instances that have been removed from $S$ have no associates; besides, sorts $S$ in an attempt to remove the central points before the border points. However, noisy instances can also be border points, which can cause a change in the order of removal of instances and some of these can remain in $S$ even after the noisy instance is removed.

DROP3. This algorithm uses a noise-filtering before sorting the instances in $S$, That is done using a rule similar to ENN that eliminates any misclassified instance by its $k$ nearest neighbors. Also, it removes noisy instances, as well as close border points. This helps to avoid overfitting the data.

DROP4. The noise-filtering of this algorithm pass removes each instance only if, 1) It is misclassified by its k nearest neighbors, and 2) Its removal does not affect the classification of other instances.

DROP5. This algorithm modifies DROP2 so, that instances are considered for removal beginning with instances that are closest to their nearest enemy, and proceeding to outside.

The Decremental Encoding Length algorithm (DEL) is similar to DROP3, except that it uses the length encoding heuristic to decide in each case whether the instance can be removed or not. In this DEL algorithm an instance is removed only if a) It is misclassified by its k nearest neighbors, and b) The removal of the instance does not increase the encoding length cost.

### Class Conditional Instance Selection (CCIS)

Marchiori [61] developed a large margin-based algorithm for instance selection, where the effect of eliminating one instance on the hypothesis margin of other

instances is measured by the concept of class conditional nearest neighbor. The hypothesis margin is defined as the distance between the hypothesis and the closest hypothesis that assigns alternative label to the given instance. The proposed instance selection method can be interpreted as a novel large-margin-based procedure for training Voronoi networks.

### Random Mutation Hill Climbing Algorithm (RMHC)

Skalak [85] used the wrapper approach for feature selection and for decreasing the number of prototypes stored in instance-based methods. The general approach is to use a bit string to represent a set of prototypes, and in some experiments, a collection of features. The intuitive search mechanism is that the mutation of the bit vector changes the selection of instances in the prototype set or toggles the inclusion or exclusion of a feature from the nearest neighbor computation.

### Steady-State MA (SSMA)

Garcia *et al.* [37] proposed a model of memetic algorithm for instance selection, that incorporates an ad-hoc local search specifically designed for optimizing the properties of prototype selection problem with the aim of tackling the scaling up problem. Memetic algorithms combine evolutionary algorithms and local search within the evolutionary cycle.

# 5

# Contributions

In this chapter techniques to improve generalization and to prevent the induction of overly complex models in the context of nearest neighbor condensing classification techniques are introduced.

With this aim, we relax the notion of consistency of a subset by introducing the notion of $\alpha$-consistent subset ($\alpha \in [0, 1]$), that is a subset that correctly classifies at least the $\alpha$ fraction of the training set. Then we describe a variant of the FCNN algorithm [8], called $\alpha$-FCNN rule, that computes an $\alpha$-consistent subset (see Section 5.1)

We then introduce some subset selection strategies, namely PACOPT, MAXOPT, and TRNOPT (see Section 5.2), intended to select the most promising subset according to different ways of estimating expected accuracy. Among them, the PACOPT strategy is based on PEE principle and estimates generalization as a trade-off between training set accuracy and model complexity.

Moreover, a variant of the PACOPT strategy, called aPACOPT for approximate-PACOPT, is described (see Section 5.2.4). The technique attempts to reduce time complexity by early terminating the learning phase on the basis of the current trend of the pessimistic accuracy estimate curve.

Before going into the details, next we summarize the contributions of this research:

- As the first major result (see Section 5.3), we show that the PACOPT selection strategy guarantees to preserve the accuracy of the consistent subset with a larger reduction factor, since on the average the subset selected by PACOPT contains the 30% of the training set consistent subset objects.
- As the second major result, we show that a sensible, on the average of the 2%, generalization improvement can be obtained by using a reduced subset of intermediate size, consisting on the average of the 63% of the training set consistent subset objects.

- Moreover, the aPACOPT strategy allows to compute approximatively the same model determined by the PACOPT strategy, but with sensibly smaller time requirements (the execution time of aPACOPT corresponds to the $25\% - 50\%$ of the time required by PACOPT).
- Comparison with state-of-the-art prototype selection methods highlight that FCNN-PAC strategies are able to obtain a model of size comparable to that obtained by the best prototype selection methods in terms of reduction ratio, with far smaller time requirements, corresponding to four orders of magnitude on medium-sized datasets.

## 5.1 The $\alpha$-FCNN algorithm

We start by giving some preliminary definitions.

In the following we denote by $T$ a labeled training set from a metric space with distance metrics d.

We denote by $nn(p, T)$ the nearest neighbor of $p$ in $T$ according to the distance d. If $p \in T$, then $p$ itself is its nearest neighbor. We denote by $l(p)$ the label associated with $p$.

Given a point $q$, the NN rule $NN(q, T)$ assigns to $q$ the label of the nearest neighbor of $q$ in $T$, i.e. $NN(q, T) = l(nn(q, T))$.

A subset $S$ of $T$ is said to be a *training set consistent subset of* $T$ if, for each $p \in (T - S)$, $l(p) = NN(p, S)$.

Now we relax the notion of training set consistent subset.

**Definition 5.1.** *Let $\alpha$ be a real value in $[0, 1]$, also called consistency fraction. A subset $S$ of $T$ is said to be training set $\alpha$-consistent subset of $T$ if it correctly classifies at least the $\alpha$ fraction of the objects in $T$, that is to say if*

$$\frac{|\{p \in T : l(p) = NN(p, S)\}|}{|T|} \geq \alpha$$

□

Thus, a training set consistent subset corresponds to a training set 1-consistent subset, and vice versa.

The $\alpha$-FCNN algorithm (for $\alpha$-consistent Fast Condensed Nearest Neighbor rule) is a specialization of the FCNN algorithm [7, 8] for computing an $\alpha$-consistent subset.

Let $S$ be a subset of $T$ and let $p$ be an element of $S$. We denote by $Vor(p, S, T)$ the set $\{q \in T \mid p = nn(q, S)\}$, that is the set of the elements of $T$ that are closer to $p$ than to any other element $p'$ of $S$. Furthermore, we denote by $Voren(p, S, T)$ the set of the *Voronoi enemies* of $p$ in $T$ w.r.t. $S$, defined as $\{q \in Vor(p, S, T) \mid l(q) \neq l(p)\}$.

We denote by $Centroids(X)$ the set containing the centroids[1] of the objects in $X$.

The following Theorem states the property exploited by the FCNN rule in order to compute a training set consistent subset.

**Theorem 5.2.** *$S$ is a training set $\alpha$-consistent subset of $T$ for the nearest neighbor rule if and only if*

$$\left(1 - \frac{1}{|T|} \sum_{p \in S} |\, Voren(p, S, T)|\right) \geq \alpha$$

*Proof.* The elements of $Voren(p, S, T)$ are precisely the misclassified objects belonging to the Voronoi cell induced by the object $p$. Thus, the summation above corresponds to the number of training examples that are misclassified by using $S$ as reference set for the nearest neighbor rule, while the left hand side represents the accuracy of the subset $S$. The result then follows by noticing that the accuracy is not smaller than $\alpha$. □

The algorithm $\alpha$-FCNN is reported in Figure 1. It starts by selecting the centroid of the most populated class.

Then it works in an incremental manner: during each iteration the set $S$ is augmented with $\Delta S$ until the stop condition, given by Theorem 5.2, is reached.

The set $\Delta S$ is composed of one single object per iteration. Specifically, $\Delta S$ is built by selecting $rep(p^*, Voren(p^*, S, T))$, that is the representative of the Voronoi enemies of one of the elements of $S$. Such an element can be defined as the object $p^*$ of $S$ such that $|\, Voren(p^*, S, T)|$ is maximum, that is, the object inducing the Voronoi cell containing the maximum number of misclassified points. Note that if at the end of a generic iteration $p^*$ is undefined then the algorithm computed a 1-consistent subset.

The representative $rep(p, X)$ of $X$ w.r.t. $p$ is defined as the class centroid in $X$ closest to $p$, that is $rep(p, X) = nn(p, Centroids(X))$ (in this case $\alpha$-FCNN selects $nn(p^*, Centroids(\, Voren(p^*, S, T)))$).

In the following the latter definition of representative is employed since this variant exhibits the greatest area under the curve of the training set accuracy versus the current subset size.

## 5.2 Selection strategies

In this section we describe the strategies we have designed in order to select the FCNN condensed set associated with the best generalization power.

[1] In the Euclidean space, the centroid of a class is the object of the class closest to the geometric center of the class. In a metric space, it can be defined as the object of the class minimizing the sum of distances to any other object of the same class.

**Algorithm 1:** The $\alpha$-FCNN rule.

**Input:** A training set $T$, a consistency threshold $\alpha$;
**Output:** A training set $\alpha$-consistent subset $S$ of $T$;

**Method:**
$i = 0$;
$S_0 = \emptyset$;
$\ell^* = \arg\max_\ell |\{p \in T : l(p) = \ell\}|$;
$\Delta S_0 = \mathit{Centroids}(\{p \in T : l(p) = \ell^*\})$;
$\alpha_0 = |\{p : l(p) = \ell^*\}|/|T|$;
**while** $(\alpha_i < \alpha)$ **do**
  $i = i + 1$;
  $S_i = S_{i-1} \cup \Delta S_{i-1}$;
  $\Delta S_i = \emptyset$;
  $e_i = 0$;
  $err_{\max} = 0$;
  $p^* = \mathit{undef}$;
  **foreach** $(p \in S)$ **do**
    $err = |\mathit{Voren}(p, S, T)|$;
    **if** $(err > err_{\max})$ **then**
      $p^* = p$;
      $err_{\max} = err$;
    $e_i = e_i + err$;
  **if** $(p^* \neq \mathit{undef})$ **then**
    $\Delta S_i = \{\mathit{rep}(p^*, \mathit{Voren}(p^*, S, T))\}$;
  $\alpha_i = 1 - e_i/|T|$;
**return**$(S_i)$;

### 5.2.1 TRNOPT

According to the first technique, named TRNOPT (for *Optimal Training Accuracy* estimate), the accuracy estimate corresponds to the training accuracy. This means that the method selects as the optimal subset the (1−)consistent subset $S$ of the training set. With this aim we run the $\alpha$-FCNN algorithm with $\alpha = 1$ and then use the computed subset as the classification model.

### 5.2.2 PACOPT

Prepruning is a standard technique for handling noise and preventing overfitting in decision tree learning. Prepruning heuristics are used to reduce (not entirely eliminate) the amount of overfitting, so that both learning and model building will be more efficient. The idea of the second strategy is similar to the approach of prepruning used during decision trees growth.

The *Pessimistic Error Estimate* approach [91] computes generalization error of a classifier $C$ as the sum of the associated training error and of a

*penalty* term. The latter term intends to take into account model complexity. The resulting error $e_{pess}(C)$, which can be considered the pessimistic error estimate of the generalization error of $C$, is hence computed as follows:

$$e_{pess}(C) = \frac{e(C) + \Omega(C)}{N_{train}},$$

where $e(C)$ is the number of misclassified training examples (or absolute training error) of the classifier $C$, $N_{train}$ is the number of training examples, and $\Omega(C)$ is the penalty term associated with $C$.

In the case of decision trees, $\Omega(C)$ can be computed as $N_{leafs}(C) \cdot p$, where $N_{leafs}(C)$ is the number of leafs of the tree $C$ and $p$ is a penalty factor, that is to say a constant accounting for the complexity cost associated with a single leaf node. Hence, $p = 0.5$ means that a node should always be expanded into its two child nodes as long as it improves the classification of at least one training record, because expanding a node, which is equivalent to adding 0.5 to the overall error, is less costly than committing one training error. For example, if $e(C) = 4$, $N_{train} = 24$, $N_{leafs}(C) = 7$, and $p = 0.5$, we obtain

$$e_{pess}(C) = \frac{4 + 7 \cdot 0.5}{24} = 0.3125$$

Suppose now that by expanding a leaf node of $C$ we obtain the novel tree $C_{new}$ such that $e(C_{new}) = 2$, then

$$e_{pess}(C_{new}) = \frac{2 + 8 \cdot 0.5}{24} = 0.25$$

Since $e_{pess}(C_{new}) < e_{pess}(C)$ we can conclude that $C_{new}$ is preferable to $C$.

The PACOPT (for *Optimal Pessimistic Accuracy* estimate) strategy incorporates model complexity into the evaluation of the training set $\alpha$-consistent subset by exploiting pessimistic error estimate. Specifically, let $\alpha_i$ the training accuracy of the training set consistent subset $S_i$ determined at the beginning of the $i$-th iteration of the FCNN algorithm. According to the PACOPT criterion, the selected subset $S^*$ is the one maximizing the pessimistic accuracy estimate determined as follows:

$$\begin{aligned} &1 - pessimistic\_error\_estimate = \\ &= 1 - \frac{misclassified\_examples + penalty}{total\_examples} = \\ &= 1 - \frac{(1 - \alpha_i)|T| + p \cdot |S_i|}{|T|} = \\ &= \alpha_i - p \cdot \frac{|S_i|}{|T|}, \end{aligned}$$

where the penalty term associated with the subset $S_i$ is $p \cdot |S_i|$.

Differently from decision trees, here we assume that the penalty term is given by $p \cdot |S|$, the product of the penalty factor by the number of elements the $\alpha$-consistent subset, since for the nearest neighbor decision rule all the reference objects play the same role (loosely speaking, they can be regarded as the leafs of a "flat" tree).

Thus,

$$S^* = \arg\max_{S_i} \left\{ \alpha_i - p \cdot \frac{|S_i|}{|T|} \right\}$$

The penalty factor $p$ intends to model the cost of including another object in the subset $S_i$. This corresponds to say that the subset $S_j$ (having training accuracy $\alpha_j$) is preferable to $S_i$ (having training accuracy $\alpha_i \leq \alpha_j$) provided that:

$$\begin{aligned} \alpha_i - p \cdot \frac{|S_i|}{|T|} < \alpha_j - p \cdot \frac{|S_j|}{|T|} &\iff \\ \alpha_i - p \cdot \frac{|S_i|}{|T|} < \alpha_j - p \cdot \frac{|S_i| + |S_j - S_i|}{|T|} &\iff \\ \alpha_i < \alpha_j - p \cdot \frac{|S_j - S_i|}{|T|} &\iff \\ (\alpha_j - \alpha_i) \cdot |T| > p \cdot |S_j - S_i| & \end{aligned}$$

Intuitively, this means that the methods prefers a larger subset provided that the number of further correctly classified examples overcomes by a factor $p$ the number of additional examples to be included in the model.

Applying the PACOPT criterion does not require additional asymptotic operations and, hence, its cost is the same of the TRNOPT criterion.

### 5.2.3 MAXOPT

The last strategy, named MAXOPT (for *Optimal Cross Validation Accuracy* estimate) exploits cross validation in order to select the model showing the best generalization performance.

With this aim, the parameter $\alpha$ is varied within a suitable interval $[\alpha_{\min}, 1]$ by considering values $\alpha_1 = \alpha_{\min}, \alpha_2, \ldots, \alpha_m = 1$ ($m > 1$). For each $\alpha_j$ ($1 \leq j \leq m$) a stratified ten-fold cross validation is performed in order to determine the average accuracy associated with the training set $\alpha_j$-consistent subset. The best consistency fraction $\alpha^*$ is the one associated with the maximum average cross validation accuracy. The optimal model $S^*$ for MAXOPT is eventually obtained as the $\alpha^*$-consistent subset extracted from the whole training set.

### 5.2.4 The aPACOPT selection strategy

The PACOPT strategy aims at improving model size, while preserving or even improving accuracy, over the TRNOPT strategy, but presents the same temporal cost due the need of computing the whole pessimistic accuracy estimate curve.

In this section, a variant of the PACOPT strategy, called aPACOPT for approximate-PACOPT, is described. The technique attempts to reduce time complexity by early terminating the learning phase on the basis of the current trend of the pessimistic accuracy estimate curve.

The termination condition of aPACOPT assumes that the above curve is monotone non-decreasing before the local maximum and monotone non-increasing after the local maximum. However, the design of the termination condition is made difficult by the fact that the learning curve is not regular (i.e. monotone), but instead presents a lot of local fluctuations. In order to make the termination condition robust to such a fluctuations, the termination will be based on the behavior of the curve within a window of size $w$.

Specifically, let $t = 1, 2, \ldots, |S|$ denote the current iteration of the algorithm, also referred to as time $t$ in the following. The window $W^{(t)}$ at time $t \geq w$ is the pair $\langle X^{(t)}, Y^{(t)} \rangle$, where $X^{(t)}$ is a vector of $w$ elements (or timestamps):

$$X^{(t)} = (X_1^{(t)} = t - w + 1,\ X_2^{(t)} = t - w + 2, \ldots,\ X_w^{(t)} = t),$$

representing the time interval $[t - w + 1, t]$ and $Y^{(t)} = (Y_1, ..., Y_w)$ is a vector of $w$ elements representing the pessimistic accuracy estimate:

$$Y_i^{(t)} = \alpha_{t-w+i} - p \cdot \frac{|S_{t-w+i}|}{|T|},$$

within time interval $[t-w+1, t]$. The vector $\hat{Y}^{(t)}$ consists of the $t-w$ accuracy estimates

$$\hat{Y}^{(t)} = \left( \alpha_1 - p \cdot \frac{|S_1|}{|T|},\ \alpha_2 - p \cdot \frac{|S_2|}{|T|}, \ldots,\ \alpha_{t-w} - p \cdot \frac{|S_{t-w}|}{|T|} \right)$$

pertaining to the windows that precede the current one.

The current window can be exploited to approximate the slope tangent line to the pessimistic accuracy curve at time $t$ by computing the intercept $\beta_t$ of the regression line for the data points $\{(X_i^{(t)}, Y_i^t), i = 1, \ldots, w\}$, as follows

$$\beta_t = \frac{\sum_{i=1}^{w} \left[ (X_i^{(t)} - \overline{X}^{(t)})(Y_i^{(t)} - \overline{Y}^{(t)}) \right]}{\sum_{i=1}^{w} (X_i^{(t)} - \overline{X}^{(t)})},$$

where $\overline{X}^{(t)}$ ($\overline{Y}^{(t)}$, resp.) denotes the mean of the elements in $X^{(t)}$ ($Y^{(t)}$, resp.).

Let us define two boolean variables, that are

$$\varphi_t = \begin{cases} \textbf{true} & \text{, if } \beta_t < \epsilon \\ \textbf{false} & \text{, otherwise} \end{cases}$$

representing the fact that the slope of curve tangent is non-increasing (where $\epsilon$ is a small positive value, e.g. $\epsilon = 10^{-3}$, intended to capture errors associated with the estimation procedure), and

$$\psi_t = \begin{cases} \textbf{true} & \text{, if } \max(Y^{(t)}) < \max(\hat{Y}^{(t)}) \\ \textbf{false} & \text{, otherwise} \end{cases}$$

representing the fact that the global maximum has not been observed in the current window.

Now we are in the position of providing the termination condition of the aPACOPT strategy. The method stops if the following condition holds:

$$STOP_t \equiv \left( \bigwedge_{i=1}^{w} \varphi_{t-w+i} \right) \wedge \left( \bigwedge_{i=1}^{w} \psi_{t-w+i} \right),$$

which can be interpreted as the fact that the slope of curve has been non-increasing and no new local maximum has been observed during an entire window.

At the time $t$ at which aPACOPT decides to stop, the best pessimistic accuracy estimate $\tilde{\alpha}^*$ so far encountered is determined and the associate subset is returned as the solution. It can be concluded that the subset returned by aPACOPT cannot be greater than those computed by PACOPT, since two cases are possible: (1) $\tilde{\alpha}^* < \alpha^*$, and this means that the algorithm stopped before reaching the maximum $\alpha^*$, which is associated with a larger subset; or (2) $\tilde{\alpha}^* = \alpha^*$, and this means that the algorithm stopped after reaching the maximum and, hence, selected the subset associated with $\alpha^*$.

## 5.3 Experimental Results

In this section we present experimental results involving the introduced techniques.

A number of training sets are from the UCI Machine Learning Repository[2], whose characteristics are summarized in Table 5.1 where, for each data set, the name, abbreviation, size, features, and number of classes are given. Moreover, we employed also the MNIST dataset[3] which consists of handwritten digits (60,000 objects, 784 features, and 10 classes) and the Forest Cover Type dataset (Covtype, for short, 600,000 objects, 54 features, and 7 classes).

[2] `http://archive.ics.uci.edu/ml/`

[3] `http://yann.lecun.com/exdb/mnist/`

| *Data set name* | *Abbrev.* | *Size* | *Features* | *Classes* |
|---|---|---|---|---|
| Bupa | BUP | 345 | 6 | 2 |
| Coil 2000 | COI | 4,992 | 85 | 2 |
| Colon Tumor | COL | 62 | 2,000 | 2 |
| Echocardiogram | ECH | 61 | 11 | 2 |
| Ionosphere | ION | 351 | 34 | 2 |
| Iris | IRI | 150 | 4 | 3 |
| Image Segmentation | IMA | 2,310 | 19 | 7 |
| Pen Digits | PEN | 7,494 | 16 | 10 |
| Pima Indians Diabetes | PIM | 768 | 8 | 2 |
| Satellite Image | SAT | 6,435 | 36 | 6 |
| Spam Database | SPA | 4,207 | 57 | 2 |
| SPECT Heart Data | SPE | 349 | 44 | 2 |
| Vehicle | VEH | 846 | 18 | 4 |
| Wisconsin Breast Cancer | WBC | 683 | 9 | 2 |
| Wine | WIN | 178 | 13 | 3 |
| Wisc. Progn. Breast Cancer | WPB | 198 | 33 | 2 |

Table 5.1: Data Sets Used in the Experiments.

The MNIST dataset, which contains binary images of handwritten digits, is commonly used for training various image processing systems. All digit images have been size-normalized and centered in a fixed size image of $28 \times 28$ pixels. In the original dataset each pixel of the image is represented by a value between 0 and 255, where 0 is black, 255 is white and anything in between is a different shade of grey.

The Forest Cover Type dataset contains tree observations from four wilderness areas of the Roosevelt National Forest in Colorado. All observations are cartographic variables (no remote sensing) from $30 \times 30$ meter sections.

All the experiments were executed on a personal computer based on an Intel Core i7-6700 3.40GHz processor, equipped with 16GB of RAM, and under Linux Operating System.

### 5.3.1 Accuracy of the strategies

Here, we study the accuracy and model size of the PACOPT, TRNOPT, and MAXOPT strategies by exploiting the dataset described in Table 5.1. As for the aPACOPT strategy, since it is tailored on large datasets, we defer its analysis to the following section devoted to the scalability analysis and to the comparison with competitors methods.

In order to assess performances of the strategies and to measure generalization, for PACOPT and TRNOPT ten fold cross validation has been accomplished and the average values over the ten folds are reported, while MAXOPT has been executed on the whole training set by considering the same folds. Thus, in these experiments accuracy associated with MAXOPT

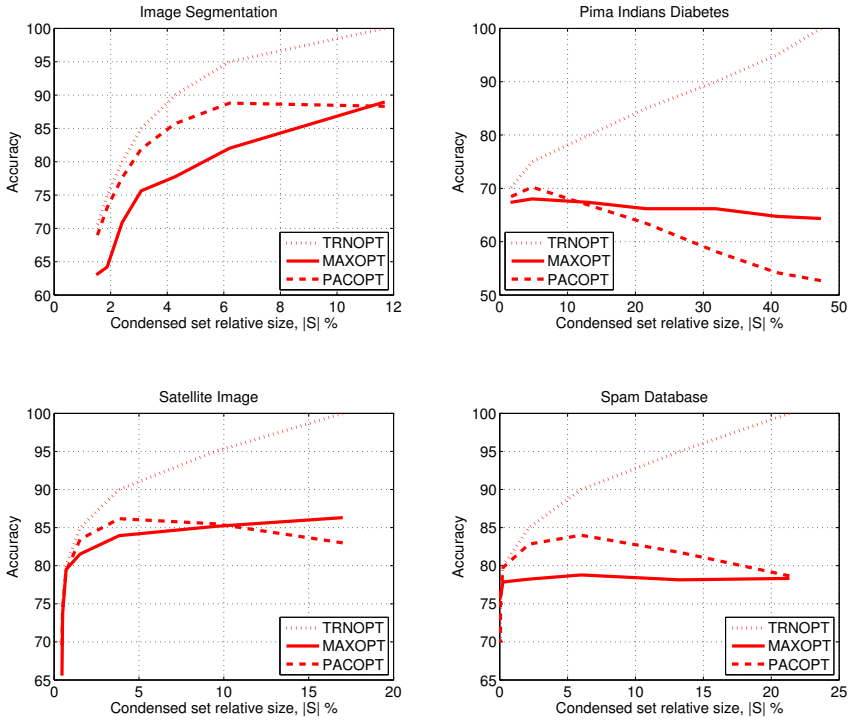



Fig. 5.1: TRNOPT, PACOPT, and MAXOPT accuracy curves.

represents the best generalization achievable by an $\alpha$-consistent subset for the nearest neighbor rule.

**Accuracy curves.** First of all, in order to make clear the behavior of the subset selection strategies, we show in Figure 5.1 the accuracy estimated by the three methods on the subsets $S_i$ associated with the various accuracy levels $\alpha_i$.

As far as TRNOPT is concerned, the estimated accuracy is equal to the training one and, hence, the associated curve is always increasing and reaches its optimum at the $\alpha_{\max} = 1$ accuracy level, which corresponds to the largest subset (the 1-consistent one).

As far as PACOPT is concerned, since its estimated accuracy depends on the trade-off between the training accuracy and the penalty term taking into account subset size, the associated curve is firstly increasing and then can be decreasing if the peak is reached before 100% training accuracy. For this strategy, the optimal subset is determined in correspondence of the estimated accuracy peak. For example, on *Spam* the PACOPT estimate increases up to $|S| = 6\%$ and then decreases, while TRNOPT has its maximum at $|S| \approx 21\%$.

As far as MAXOPT is concerned, in general its curve depends on cross-validation accuracy and no specific trend is guaranteed. Since in the experiments presented we are using cross validation accuracy as a measure of generalization, in this case the optimal subset is that in correspondence of the maximum of the MAXOPT accuracy estimate curve.

| | PACOPT | | | | MAXOPT | | | | TRNOPT | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Acc* | *ΔAcc* | *Size* | *ΔSize%* | *Acc* | *ΔAcc* | *Size* | *ΔSize%* | *Acc* | *Size* |
| BUP | **63.53** | +4.71 | 6.44 | −88.23 | **63.53** | +4.71 | 6.44 | −88.23 | 58.82 | 54.75 |
| COI | **94.25** | +10.16 | 0.02 | −99.90 | **94.25** | +10.16 | 0.02 | −99.90 | 84.09 | 20.47 |
| COL | **80.00** | 0.00 | 3.58 | −90.01 | **81.67** | +1.67 | 17.92 | −50.00 | 80.00 | 35.84 |
| ECH | **93.33** | 0.00 | 5.46 | −50.04 | 93.33 | 0.00 | 10.93 | 0.00 | 93.33 | 10.93 |
| ION | 85.14 | −0.29 | 5.70 | −69.98 | **87.71** | +2.28 | 12.03 | −36.65 | 85.43 | 18.99 |
| IRI | 92.00 | −2.00 | 2.96 | −71.46 | 94.00 | 0.00 | 10.37 | 0.00 | 94.00 | 10.37 |
| IMA | 82.03 | −6.93 | 6.20 | −46.96 | 88.96 | 0.00 | 11.69 | 0.00 | 88.96 | 11.69 |
| PEN | **98.42** | 0.00 | 3.71 | 0.00 | 98.42 | 0.00 | 3.71 | 0.00 | 98.42 | 3.71 |
| PIM | **68.03** | +3.69 | 4.77 | −88.45 | **68.03** | +3.69 | 4.77 | −88.45 | 64.34 | 41.31 |
| SAT | 83.95 | −2.36 | 3.83 | −77.48 | 86.31 | 0.00 | 17.01 | 0.00 | 86.31 | 17.01 |
| SPA | **78.79** | +0.46 | 6.00 | −71.88 | **78.79** | +0.46 | 6.00 | −71.88 | 78.33 | 21.34 |
| SPE | 70.29 | −16.18 | 10.19 | −65.95 | 86.47 | 0.00 | 29.93 | 0.00 | 86.47 | 29.93 |
| VEH | 59.40 | −3.76 | 12.74 | −74.80 | 63.10 | 0.00 | 50.56 | 0.00 | 63.10 | 50.56 |
| WBC | **95.74** | +1.92 | 0.33 | −96.17 | **95.74** | +1.92 | 0.33 | −96.17 | 93.82 | 8.62 |
| WIN | **68.24** | 0.00 | 17.48 | −55.56 | 68.24 | 0.00 | 39.33 | 0.00 | 68.24 | 39.33 |
| WPB | **66.32** | +2.11 | 3.37 | −92.67 | **74.32** | +10.11 | 0.56 | −98.78 | 64.21 | 46.02 |
| $\mu$ | *80.70* | *−0.61* | *5.66* | *−70.76* | *83.37* | *+2.06* | *13.59* | *−37.06* | *81.31* | *25.31* |
| $\sigma$ | *±12.66* | *±5.47* | *±4.34* | *±24.16* | *±11.92* | *±3.36* | *±14.10* | *±43.48* | *±12.89* | *±16.35* |

Table 5.2: Accuracy and Model Size.

**Accuracy and model size.** Table 5.2 summarizes the accuracy and model size achieved by the strategies on the whole set of datasets.

The *Size* columns report the relative size of the subset computed by each method (values are expressed in percentage of the size of the whole dataset; thus, e.g., $Size = 6.44$ for PACOPT on dataset BUP means that the subset is composed of the 6.44% of the dataset objects).

The $\Delta Size$ columns report the variation of the size of the subset computed by PACOPT and MAXOPT methods with respect to the size computed by TRNOPT method. Specifically, values are obtained as follows:

$$\Delta Size\%_{\text{METHOD}} = 100\left(1 - \frac{Size_{\text{METHOD}}}{Size_{\text{TRNOPT}}}\right)$$

The *Acc* columns report the Optimal Pessimistic Accuracy Estimate (PACOPT), Optimal Cross Validation Accuracy (MAXOPT), Optimal Training accuracy (TRNOPT).

As for PACOPT is concerned, accuracies are highlighted in bold when their value is greater or equal than that obtained by TRNOPT.

As for MAXOPT is concerned, due to the experimental design this method scores always the maximum accuracy. Hence, accuracy values are highlighted in bold when they are strictly greater than that obtained by TRNOPT.

The $\Delta Acc$ columns report the variation of accuracy of PACOPT and MAXOPT with respect to TRNOPT:

$$\Delta Acc = Acc_{\text{METHOD}} - Acc_{\text{TRNOPT}}$$

**Assessing performances.** In order to assess performances of the strategies, we performed the Wilcoxon signed rank test for zero mean. It is a test of the hypothesis that the difference $x_i - y_i$ between the pairs of samples in the vectors $x$ and $y$ comes from a distribution whose mean is zero.

Recall that the $p$-value is the probability of obtaining a test statistic value at least as extreme as the one that was actually observed, assuming that the null hypothesis (in our case, that the observed distribution complies with the theoretical one) holds. Before the test is performed, a threshold value is chosen, called the significance level $\lambda$ of the test, traditionally 5% ($\lambda = 0.05$) or 1% ($\lambda = 0.01$).

In the specific case, the null hypothesis "mean is zero" can be rejected at the $\lambda$ level provided that the $p$-value is smaller than $\lambda$.

Table 5.3 reports the $p$-values of the Wilcoxon test for zero mean of the difference of accuracy by considering accuracy values reported in Table 5.2.

As far as MAXOPT is concerned, $p$-values are reported in order to confirm that the size of the sample and the number of wins are large enough to reach a robust significance level. And, indeed, it can be seen that for MAXOPT the null hypothesis can be always rejected at significance level 1%.

| | PACOPT | TRNOPT |
|---|---|---|
| MAXOPT | 0.004 | 0.008 |
| PACOPT | — | 0.787 |

Table 5.3: $p$-values of the Wilcoxon test for zero mean of the difference of accuracy of the three strategies.

| | PACOPT | MAXOPT |
|---|---|---|
| BUP | 0.012 | 3.00 |
| COI | 0.480 | 76.64 |
| COL | 0.008 | 5.32 |
| ECH | 0.004 | 1.32 |
| ION | 0.016 | 2.24 |
| IRI | 0.004 | 1.28 |
| IMA | 0.036 | 11.52 |
| PEN | 0.088 | 31.48 |
| PIM | 0.020 | 4.76 |
| SAT | 0.332 | 88.88 |
| SPA | 0.444 | 98.88 |
| SPE | 0.016 | 3.8 |
| VEH | 0.024 | 9.08 |
| WBC | 0.012 | 1.64 |
| WIN | 0.008 | 1.76 |
| WPB | 0.008 | 2.40 |

Table 5.4: Execution Times (seconds).

As a valuable result, the $p$-value of the test for the PACOPT and TRNOPT strategies is far larger than an acceptable rejecting level. Hence, from the point of view of the guaranteed accuracy the two methods are comparable.

Putting all things together, it can be concluded that *PACOPT guarantees to preserve the accuracy of the training set consistent subset, but with a far smaller subset*. On the average, the reduction factor over the 1-consistent subset is of the 70%. As for MAXOPT, the experiments highlights that by selecting a reduced condensed subset, sensible accuracy improvements can be obtained, on the average the 2%, with an appreciable reduction of the subset, on the average the reduction factor associated with the optimal subset is of the 37%, which corresponds about to half of the reduction factor of PACOPT for the parameter setting here considered.

**Execution time.** For completeness, Table 5.4 presents the average time elapsed (in seconds) to complete a run of the method on the datasets above considered. Results highlight that MAXOPT is more demanding in terms of CPU usage than PACOPT, due to the need to process all the different folds.

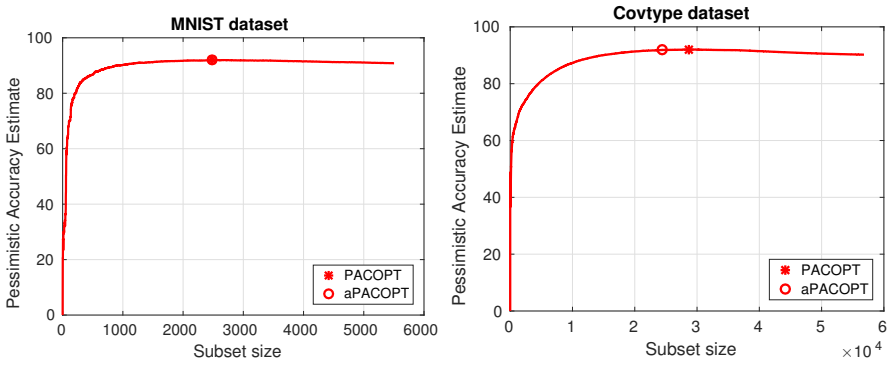



Fig. 5.2: Pessimistic Accuracy Estimate curves.

### 5.3.2 Analysis of the aPACOPT strategy

The rationale of the aPACOPT strategy is to compute a model comparable to that determined by PACOPT, but with sensibly smaller time requirements. This section is devoted to experimentally substantiating the above claim in the context of large datasets.

Figure 5.2 shows the trend of the Pessimistic Accuracy Estimate of the aPACOPT strategy for the MNIST and Covtype datasets.

It can be noticed that the trend of the curves agrees with the assumption the termination condition of the aPACOPT strategy is designed on, since it resembles a concave function.

As for the subset selected by aPACOPT, it can be seen that in both case it is of remarkable quality if compared to that selected by PACOPT. Indeed, on MNIST the model size of PACOPT and aPACOPT coincide (model size 2,485, 4.1% of the dataset size, pessimistic accuracy estimate 91.97%), while on Covtype the model of aPACOPT (model size 24,327 corresponding to the 4.2% of the dataset size and scoring a pessimistic accuracy estimate of 91.87%) is slightly smaller than that of PACOPT (model size 28,651 corresponding to the 4.9% of the dataset size and scoring a pessimistic accuracy estimate of 92.05%).

Differences are justified by the fact that aPACOPT could get stuck in a local maximum since, although from a macroscopic point of view the learning curve appears to be monotone non-decreasing before the global maximum and monotone non-increasing after, from a microscopic point of view it presents local fluctuations.

Notice that the size of the model selected by TRNOPT is $56,802$ (9.8% of the dataset size) for Covtype and 5,503 (9.2% of the dataset size) for MNIST. Hence, in both cases the size of the model selected by aPACOPT is more than halved with respect to the basic strategy.

Figure 5.3 accounts for the growth of the model size as a function of the dataset size. Specifically, the plots report the size of the model as a percentage

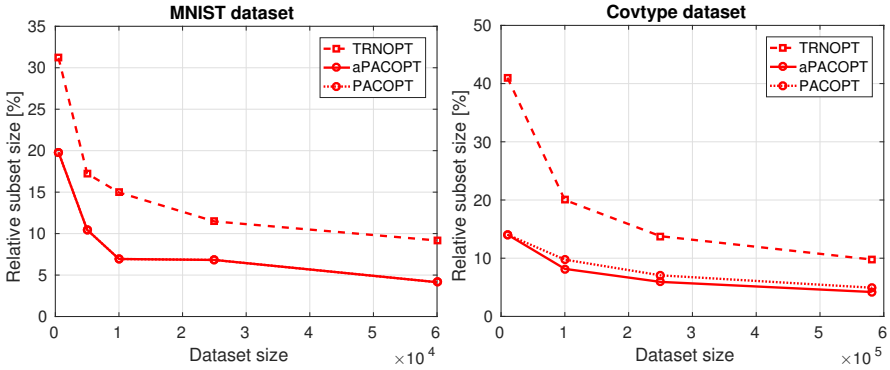



Fig. 5.3: TRNOPT, PACOPT, and aPACOPT relative model size on large datasets.

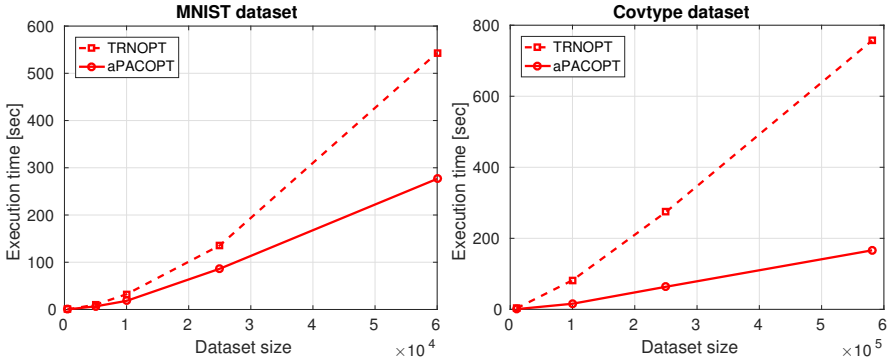



Fig. 5.4: TRNOPT, PACOPT, and aPACOPT execution time on large datasets.

of the dataset size which it is induced on. It can be seen that all the strategies are particularly suitable for large data, since in both cases the percentage of the selected objects is decreasing with the number of objects composing the training set.

Finally, Figure 5.4 shows the execution time of the strategies versus the size of the dataset. We recall that the execution time of TRNOPT and PACOPT is the same. The plots highlight that, despite the fact that aPACOPT is able to determine a model of quality comparable to that of PACOPT, by using the aPACOPT strategy great time savings are obtained with respect to the non-approximate PACOPT strategy.

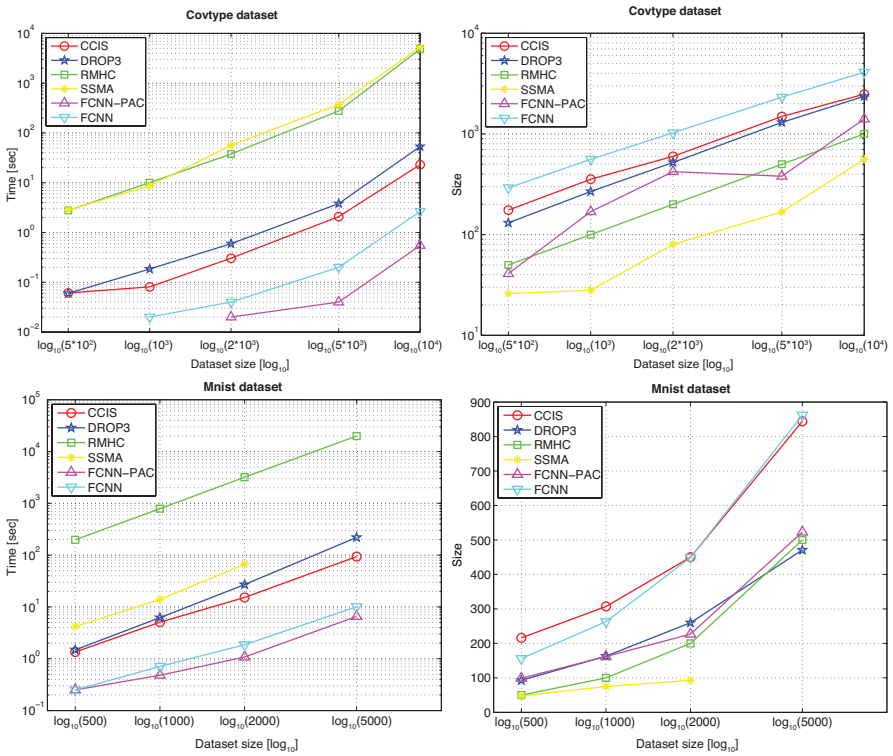



Fig. 5.5: Comparison with related methods.

### 5.3.3 Comparison with related methods

Garcia *et al.* [38] propose an extensive taxonomy based on the main characteristics presented by prototype selection methods for the nearest neighbor classification rule. The taxonomy exploits a number of criteria that can be used to evaluate the relative strengths and weaknesses of each algorithm. These include storage reduction, noise tolerance, generalization accuracy, and time requirements.

According to the analysis there accomplished, DROP3, CCIS, SSMA, and RMHC are remarkable methods belonging to the hybrid family. The best methods, considering the tradeoff reduction/accuracy rate, are RMHC, and SSMA, over medium data sets. However, these methods achieve a significant improvement in the accuracy rate at the expense of a high computational cost. The execution time of these techniques could be prohibitive when the data scales up. The methods that harm the accuracy at the expense of a great reduction of time complexity are DROP3 and CCIS.

Thus, we selected these remarkable prototype selection techniques in order to accomplish a scalability comparison taking into account the novel PAC-

based FCNN pruning strategy. As for the competitor methods we used the implementation within the KEEL software platform. KEEL[4] (Knowledge Extraction based on Evolutionary Learning) tool is an open source (GPLv3) software based on data flow to design experiments with different datasets and computational intelligence algorithms.

Figure 5.5 illustrates the scalability analysis of CCIS, DROP3, RMHC, SSMA, FCNN (TRNOPT strategy) and FCNN-PAC (aPACOPT strategy) on the MNIST and Covtype datasets.

Datasets up to 10,000 objects for Covtype and up to 5,000 objects for MNIST have been considered, due to the heavy time requirements of competitor methods. As for the execution time of FCNN and FCNN-PAC on larger dataset, the reader is referred to Figure 5.4.

Consider the model size. Although FCNN, together with CCIS, presents the smaller reduction ratio, the FCNN-PAC strategy is able to obtain a model of size comparable to that of competitors that are notable for their reduction ratio, as RMHC and DROP3. As for SSMA, it appears to be able to reach the best reduction ratio.

As for as the execution time is concerned, however, SSMA exhibits very large time requirements. On Covtype it requires, together with RMHS, more than 5,000 seconds on the dataset of size 10,000. Compare this time with the 0.55 seconds required by FCNN-PAC.

On MNIST, SSMA was not able to run on datasets of size larger than 2,000 due to memory requirements (the algorithm reported an "out of memory" error of these instances). On the same dataset, FCNN-PAC required 6.55 seconds on the larger instance, while RMHC employed about 20,000 seconds to process the same input.

[4] `http://keel.es/`

# Part II

# Anomaly Detection Techniques for Large Networks

# 6

# Anomaly Detection in Networks with Temporal Information

The large use of social networks supplies a huge amount of data which provides much information about individuals and individual behaviors reflecting human relationship in the real world. Such behaviors can be model as relational structures among the actors of the social network.

Among the interesting hidden knowledge that can be mined by analyzing node behaviors, a relevant role is played by the anomaly discovery, where the aim is to find those individuals that can be considered as outliers, since they assume exceptional behaviors. The problem of finding malicious nodes in networks is of interest in many areas such as fake account detection, spammer node detection, ddos attacks in computer networks, and many others. Much work has been made to detect anomalous nodes mostly based on detecting anomalous structures around the individual [5].

However, in many scenarios, the exceptional behavior of an individual has not to be searched only in the structural composition of its neighborhood but the exceptional behavior is characterized by the temporal sequence of connection establishments. Thus, taking into account the time dimension sheds interesting lights on individuals' behaviors. As such, the approach pursued in this thesis is orthogonal to the works aimed at mining structural properties of large static networks.

Consider, for example, the individuals registered to the Facebook social network and arcs between them defined as follows: if $a$ marks $b$ as a friend there is an arc from $a$ to $b$ and, vice versa, there is an arc from $b$ to $a$ if $b$ marks $a$ as a friend. Thus, the arc from $a$ to $b$ represents that either $a$ sends a Facebook request to $b$ or $a$ accepts a Facebook request coming from $b$.

Consider, now, an individual $a$ with five hundreds friends then with five hundreds other individuals there is a connection from $a$ and a connection towards $a$. Clearly, it is not anomalous since such a number of friends is not so exceptional. But, if $a$ is always the first to send Facebook friend requests and all the five hundreds just accept this request (and, then, for any $b$ friends of $a$ the arc from $a$ to $b$ always precedes the arc from $b$ to $a$), $a$ becomes a clear outlier.

The rest of this second part is organized as follows. Chapter 7 reports an overview of the relevant background, providing specific definition of anomalies in the context of static and dynamic graphs, and describing fundamental anomaly detection techniques. Chapter 8 presents a novel technique for node anomaly detection, demonstrating its effectiveness on large networks.

# 7

# Scenario and Related Work

Social Networking Sites are experiencing a rapid growth. Many of these sites boast with millions of members using their networks on regular basis to communicate, share, create, and collaborate with others. Popular examples of these Social Networking Sites are Facebook, LinkedIn and Twitter.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior that the other dataset objects possess.

A literature review is provided to set out the main research elements. A picture is rendered of the research topic and its theoretical fundaments.

Next, we give an overview of social networks, describing their main characteristics. Then, we describe anomaly detection in networks, with a view to understand their structure and gain insights about what roles they might be fulfilling. Finally, we provide background on anomaly detection in real world applications.

## 7.1 Social Network Analysis

Social network analysis (SNA) [62] is a set of methods and applications for analyzing network data. One of the main purposes of SNA is to detect and interpret patterns of relationships between actors and to identify the impact of the social structure on the functioning of actors and networks.

Social Network Analysis is particularly well suited for understanding and determining the global structure of a social network, the distribution of actors and activities, and the strategic positions and actors.

SNA [82] tries to understand and exploit the key features of social networks in order to manage their life cycle and predict their evolution. Much research has been conducted on SNA using graph theory which provides a formal language for describing networks and their features. SNA, sometimes also referred to as structural analysis, is not a formal theory, but rather a broad strategy for investigating social structures.

Most broadly, social network analysis [99] (1) conceptualizes social structure as a network with ties connecting members and channelling resources, (2) focuses on the characteristics of ties rather than on the characteristics of the individual members, and (3) views communities as personal communities, that is, as networks of individual relations that people foster, maintain, and use in the course of their daily lives.

The defining feature of social network analysis is its focus on the structure of relationships, ranging from casual acquaintance to close bonds. It maps and measures formal and informal relationships to understand what facilitates or impedes the knowledge flows that bind interacting units (e.g., data and information, voice, or video communications). Social network analysis is a method with increasing application in the social sciences and has been applied in areas as diverse as psychology, health, business organization, and electronic communications. Recently, interest has grown in analysis of leadership networks to sustain and strengthen their relationships within and across groups, organizations, and related systems [84].

In [98] introduced social network analysis as a distinct research perspective within the social and behavioral sciences; distinct because social network analysis is based on an assumption of the importance of relationships among interacting units.

The growth of social network analysis has led many to see it as a new theoretical paradigm rather than simply as a collection of techniques. Social network analysis has also recently been linked with one particular theory: the theory of social capital [83].

The quality of relationships among and between people is an important factor to consider when working to promote strong and resilient neighborhoods or communities. The quality of these relationships is known as social capital. Social network analysis is a potential tool for participatory monitoring and evaluation as it is able to show the relationships that develop between individuals, groups and organizations over time [42].

The concepts of social network analysis developed out of a propitious meeting of social theory and application, with formal mathematical, statistical, and computing methodology.

Graphs have been widely used in social network analysis as a means of formally representing social relations and quantifying important social structural properties [98].

## 7.2 Tasks in Social Network Analysis

SNA consists of tasks that measure structural properties of networks. Social network analysis involves a variety of tasks. Next we mention two that are among the most relevant to the data mining field:

### 7.2.1 Centrality

One of the first approaches of SNA is to measure power, influence, or other individual characteristics of actors through the notion of centrality [95]. The main question of centrality is to define what makes an actor more central than another one.

The centrality is an important structural attribute of social networks. The simplest and perhaps the most intuitively obvious conception is that point centrality is some function of the degree of a point. The degree of a point, $p_i$, is simply the count of the number of other points, $pi(i \neq j)$, that are adjacent to it and with which it is, therefore, in direct contact [34].

The three main types of centrality are: degree centrality, betweenness centrality and closeness centrality [71, 63].

- *Degree Centrality:* is defined as the number of direct ties that involve a given node. For undirected ties this is simply a count of the number of ties for every actor. For directed networks, nodes can have both in-degree and out-degree centrality scores. The centrality measures how central or well connected an node is in a network. This theoretically signals importance or power and increased access to information or just general activity level and high degree centrality is generally considered to be an asset to an node.
- *Closeness Centrality:* measures how many ties are required for a particular node to access every other node in the network. The closeness centrality considers as most central the nodes that have the smallest average length of the paths linking an node to others. The measure will reach its maximum for a given network size when an node is directly connected to all others in the network and its minimum when an node is not connected to any others. This captures the intuition that short path lengths between nodes signal that they are closer to each other.
- *Betweenness Centrality:* may be defined as the number of shortest paths between alters that go through a given node. The betweenness centrality focuses on the ability of an node who frequently operate as the quickest bridge connection by means of shortest paths between any two other nodes in the network, have the ability to isolate, influence, manipulate or prevent contact between other parties. This intuitively measures the degree to which information or relationships have to flow through a particular node.

### 7.2.2 Community Detection.

Actors in a social network form groups. This task identify these communities through the study of network structures and topology. Community detection in large networks is potentially very useful [23]. Nodes belonging to a tight-knit community are more than likely to have other properties in common. Classic methods of finding communities in network borrow the idea of *graph partitioning* and *hierarchical clustering*.

Graph partitioning approaches needs to know information about the global structure of network and determine the number and size of subgroup that they want to get. The problem of graph partitioning consists in dividing the vertices in groups of predefined size, such that the number of edges lying between the groups is minimal. Hierarchical partitioning is a cluster analysis method in which the network of interest is divided into several subgroups. Hierarchical clustering is very common in social network analysis, engineering, marketing, and so on. Hierarchical clustering has the advantage that it does not require a preliminary knowledge on the number and size of the clusters [33].

The community detection problem aims at finding the optimal community assignment, from which it has closest properties (such as vertex similarity and edge betweeness) to real world-network and the best methods that can handle scalability issue. The optimal assignment refers to closely connected groups of nodes and a moderate number of different outliers. The community detection based research can be roughly categorized into four approaches. 1. Node-Centric 2. Group-Centric 3. Network-Centric 4. Hierarchical-Centric [92, 6].

- Node-Centric criteria requires each node in a group to satisfy certain properties such as *complete mutuality*, and *reachability*. Complete mutuality is a good measure of tie strength inside the subgroup, but it is a NP-Hard problem. An ideal cohesive subgroup is a *clique*. It is a maximum complete subgraph in which all nodes are adjacent to each other. Steps in complete mutuality include finding clique of size k, and then prune those nodes with k-1 degree. Reachability among nodes happens if there exist paths between those nodes. The most useful metrics for reachability are *k-clique* and *k-club*. k-clique is a maximal subgraph in which the largest geodesic distance between any two nodes is no greater than k. k-club restricts the geodesic distance within the group to be no greater than k.
- Group-Centric consider the connections inside a group as whole. The group is required to satisfy a *density-based* group requirement, while some nodes inside the group may have low connectivity. Group-centric approach does not guarantee reachability for each node in the group. It allows the degree of a node to vary, hence it is more suitable for large-scale networks. The objective of density-based group is to find the maximal quasi-clique easily. The steps for discovering communities applied as follows: 1) Search randomly a maximal quasi-clique in a sub-network. 2) Prune nodes and edges. This process is repeated until network reduced to a reasonable size so that a maximal quasi-clique can be found directly.
- Network-Centric consider the connections of the whole network. It aims is to create numbers of disjoint sets from the network. Typically, network-centric aim to optimize a criterion defined over a network partition rather than over one group. There are 5 known methods for this approach, namely: node similarity, latent space model, block model approximation, spectral clustering and modularity maximization.

*Node similarity* is defined by how similar their interactions are. Two nodes are structurally equivalent if they connect to the same set of nodes. This measure is too restrictive and rarely occurs in large-scale network. A key related concept is structural equivalence. *Latent space models* maps nodes in the network into a low-dimensional euclidean space such that similarity and distance are kept in the new space. Once the transformation is done, we begin clustering network in the low-dimensional space using methods like k-means. *Block models approximate* a given network by a block structure. The key objective of this method is to minimize the difference between an interaction matrix and a block structure. *Spectral clustering* is derived from the problem in graph partition. Cut is the total number of edges between two disjoint sets of nodes. Graph partition aims to find out a partition such that the cut is minimized. Two common variant used are *ratio cut* and *normalized cut*. Ratio cut represents the number of nodes in a community. Normalized cut represents the number of interactions inside group. *Modularity maximization* is proposed specifically to measure the strength of a community partition for real-world networks by taking into account the degree distribution of nodes.

- Hierarchical-Centric it aims is to build a hierarchical structure of communities based on network topology. There are mainly two types of hierarchical clustering: *divisive*, and *agglomerative*. The steps in divisive hierarchical clustering are 1) Partition the nodes into several smaller sets. 2) Each set is further partitioned into smaller sets. Agglomerative hierarchical clustering is the opposite of divisive methods. They initiate each node as community, and then choose two communities satisfying certain criteria such as modularity or node similarity. This process is iterated until there are no more nodes to merge. Agglomerative can be very sensitive to the node processing order and merging criteria adopted. Divisive clustering are more stable but computationally expensive. One particular metric to use is *edge betweenness*, which defined as the number of shortest paths between pairs of nodes that pass along one edge. At each iteration, it recursively removes the edges that have low edge betweenness or the weakest tie.

## 7.3 Anomaly Detection

Anomaly detection is a broad field, that has been studied within diverse research areas such statistic, pattern recognition, machine learning and data mining and having a lot of applications such as security, finance, health care, law enforcement, fraud, etc. Anomaly detection, also called outlier detection, refers to detecting patterns which do not comply with normal behaviors. In the past decade there has been a growing interest in anomaly detection in data represented as networks, or graphs. Chandola, *et al.* [18] categorises it in research areas and application domains. Numerous techniques have been devel-

oped for spotting anomalies in unstructured collections of multi-dimensional data points. As far as network anomaly detection is concerned, originally, techniques focused on anomaly detection in static graphs, which do not change and are capable of representing only a single snapshot of data. As real world networks are constantly changing, there has been a shift in focus to dynamic graphs, which evolve over time [77].

### 7.3.1 The Anomaly Detection Problem

Anomaly detection is an important problem with multiple applications, and thus has been developed for decades in various research domains. Noise detection is a topic related to anomaly detection, which consist in processing the data in order to remove unwanted noise so that the patterns in the data could be better analyzed. There are several factors that makes the anomaly detection problem increasingly very challenging [18]:

- Defining a normal region with all possible normal behavior is very difficult, because the boundary between normal and anomalous behavior is often imprecise. Thus an anomalous observation which lies close to the boundary can actually be normal, and vice-versa.
- When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make the anomalous observations appear like normal, thereby making the task of discriminating normal behavior more difficult.
- Normal behavior keeps evolving in many domains and a current notion of normal behavior might not be sufficiently representative in the future.
- The exact notion of an anomaly is different for different application domains. For example, in the medical domain a small deviation from normal in body temperature might be an anomaly, while similar deviation in stock price in the stock market domain might be considered as normal. Thus anomaly detection techniques developed for one domain could not be applicable for another domain.
- Noisy data often behaves like the actual anomalies and hence it is difficult to distinguish and remove.

### 7.3.2 Type of Anomaly

Anomalies can be classified into the following three categories:

1. **Point Anomaly:** refers to detecting an anomalous data instance with regard to the remainder of the data. This is the simplest type of anomaly and is the focus of the majority of research on anomaly detection.
2. **Contextual Anomaly:** refers to a data instance which is considered anomalous in a specific context, but not in others.
3. **Collective Anomaly:** refers to detecting anomalies in the form of a collection of data instances that behave abnormally with regard to the whole data set.

### 7.3.3 Anomaly Detection Techniques

Anomaly detection techniques can be divided according to the type of training data, in one of the following three categories:

- **Supervised anomaly detection:** These techniques operate on two phases. First the training phase and then the testing phase. The training set contains labels for both normal and abnormal samples to construct the predictive model. According to this set a model is learned in the training phase that is latter used to label the unclassified sample. There are two major issues that arise in supervised anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training data. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging.
- **Semi-Supervised anomaly detection:** Similar to the previous technique it also requires a training phase. The training set however contains only the normal class. The classic method used by such techniques is to build a model for the normal behavior, and then to use the model to identify anomalies in the test data.
- **Unsupervised anomaly detection:** These techniques do not require training set, and thus are most widely applicable. The basic assumption is that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from high false alarm rate.
  Many semi-supervised techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. Such adaptation assumes that the test data contains very few anomalies and the model learnt during training is robust to these few anomalies.

### 7.3.4 Output of Anomaly Detection

There are two possible outputs for anomaly detection algorithms, *scores* and *labels*. Scoring techniques assign a degree of outlierness to each instance in the test data, while labeling technique assign a label to each test data specifying whether the instance is anomalous or not. Scoring based anomaly detection techniques allow the analyst to use a domain specific threshold to select the most relevant anomalies. Techniques that provide binary labels to the test instances do not directly allow the analysts to make such a choice.

## 7.4 Anomaly Detection in Static Graphs

This section deals with finding anomalous entities (nodes, edges, sub-graphs) in static snapshots of graphs. There are mainly two types of anomaly detection in graph data, that are *plain graphs* and *attributed graphs*, each of which carries a different particularity.

The general definition for the anomaly detection problem for static graphs can be stated as follows:

**Definition 7.1.** *(Static-graphs anomaly detection problem)*
*Given the snapshot of a (plain or attributed) graph database, find the nodes and/or edges and/or substructures that are few and different or deviate significantly from the patterns observed in the graph [5].*

### 7.4.1 Anomalies in static plain graphs

Plain graphs consist of only nodes and edges. They do not hold more information than the graph structure to find patterns and spot anomalies. These structural patterns can be grouped further into two categories: *structure-based* patterns and *community-based* patterns.

#### Structure based methods

Akoglu *et al.* [5] categorizes these techniques as *feature-based* and *proximity-based*. The first group uses the graph structure to extract various graph centric features, like node degree and subgraph centrality, while the second group uses the graph structure to quantify the closeness of nodes in the graph to identify associations.

#### Community based methods

The aim of these methods is to find the densely connected groups of nodes in the graph and spot nodes that have connections across communities.

The community-based method for anomaly detection introduced in [89] use bipartite graphs. This method addresses two main problems: 1) neighborhood formation; 2) anomaly detection. The main idea of the authors is to use random-walks with restarts and graph partitioning of a given node.

Another method that aims to spot anomalies based on graph communities relies on matrix factorization. Matrix factorization has been used to address several problems ranging from dimensionality reduction to clustering [94, 41].

### 7.4.2 Anomalies in static attributed graphs

In case of static attributed graphs, the main goal is to exploit structures as well as the coherence of attributes of the graph to find patterns and spot anomalies. These methods can also be grouped into two: *structure-based* and *community-based* methods.

### Structure based methods

Structure based approaches identify frequent substructures in the graph that are rare structurally, i.e. connectivity-wise, as well as attribute-wise. One of the first works on attributed graph anomaly detection is introduced by [70], developed two techniques known as *Anomalous Substructure Detection* and *Anomalous Subgraph Detection*. The aim of anomalous substructure detection is to examine an entire graph, and to report unusual substructures contained within it. The main idea for finding the unusual subgraphs is to define a measure that penalizes those subgraphs containing few common substructures, making them more anomalous, it can be applied to any graph in which the nodes can be grouped in a meaningful way.

### Community based methods

These approaches aim to spot nodes in a graph, what is called community-outliers that do not exhibit the same characteristics as the others in the same community. The communities are analyzed based on both link and attribute similarities of the nodes they consist of.

One method for network outlier detection has been discussed in [36]. The algorithm, called Community Outlier Detection Algorithm (CODA), unifies both community discovery and outlier detection in a probabilistic formulation based on hidden Markov random fields.

Perozzi *et al.* [72] proposed a method called focused clustering and outlier detection in large attributed graphs namely FocusCO. The algorithm consists in three main steps. 1) inferring attribute weights, 2) extracting focused clusters, and 3) outlier detection.

## 7.5 Anomaly Detection in Dynamic Graphs

Almost all real networks are dynamic in nature and large in size. Real world graphs are constantly evolving and undergoing change to their structure. Detecting anomalies in this type of graphs is a very challenging task. As real world networks are constantly changing, there has been a shift in focus to dynamic graphs which evolve over time. This task is commonly defined as follows [5]:

**Definition 7.2.** *(Dynamic-Graph Anomaly Detection Problem)*

*Given a sequence of (plain or attributed) graphs, Find (i) the timestamps that correspond to a change or event, as well as (ii) the top-k nodes, edges, or parts of the graphs that contribute most to the change (attribution).*

### 7.5.1 Types of Anomalies

The detection and analysis of irregular patterns in dynamic networks are defined by Ranshous *et al.* [77] which proposed four types of anomalies.

1. **Anomalous Vertices:** aims to find a subset of the vertices such that every vertex in the subset has an irregular evolution compared to the other vertices in the graph. In static graphs, the single snapshot allows only intra-graph comparisons to be made. Dynamic graphs allow the temporal dynamics of the vertex to be included. A high-level definition follows.

   **Definition 7.3.** *(Anomalous vertices). Given* $\boldsymbol{G}$*, the total vertex set* $V = U_{t=1}^{T} V_t$*, and a specified scoring function* $f : V \rightarrow \mathbb{R}$*, the set of anomalous vertices* $V' \subseteq V$ *is a vertex set such that* $\forall v' \epsilon V'$*,* $\left| f(v') - \hat{f} \right| > C_0$*, where* $\hat{f}$ *is a summary statistic of the scores* $f(v)$*,* $\forall v \epsilon V$*.*

2. **Anomalous Edges:** aims to find a subset of the edges such that every edge in the subset has an irregular evolution. In a static graph, a distribution of the edge weights can be found, and each edge can be assigned a score. In dynamic graphs, two new main types of irregular edge evolution can be found: (1) abnormal edge weight evolution, and (2) appearance of unlikely edges between two vertices, formally defined as follows.

   **Definition 7.4.** *(Anomalous Edges). Given* $\boldsymbol{G}$*, the total edge set* $E = U_{t=1}^{T} E_t$*, and a specified scoring function* $f : E \rightarrow \mathbb{R}$*, the set of anomalous edges* $E' \subseteq E$ *is a edge set such that* $\forall e' \epsilon E'$*,* $\left| f(e') - \hat{f} \right| > C_0$*, where* $\hat{f}$ *is a summary statistic of the scores* $f(e)$*,* $\forall e \epsilon E$*.*

3. **Anomalous Subgraphs:** requires enumerating every possible subgraph. These anomalies are unique to dynamic networks, include communities that split, merge, disappear, and reappear frequently, or exhibit a number of other behaviors.

   **Definition 7.5.** *(Anomalous Subgraphs). Given* $\boldsymbol{G}$*, a subgraph set* $H = U_{t=1}^{T} H_t$ *where* $H_t \subseteq G_t$*, and a specified scoring function* $f : H \rightarrow \mathbb{R}$*, the set of anomalous subgraphs* $H' \subseteq H$ *is a subgraph set such that* $\forall h' \epsilon H'$*,* $\left| f(h') - \hat{f} \right| > C_0$*, where* $\hat{f}$ *is a summary statistic of the scores* $f(h)$*,* $\forall h \epsilon H$*.*

4. **Event and Change Detection:** Event detection has a much broader scope compared to the previous anomalies, aiming to identify time points that are significantly different from the rest. However provides less specific information than vertex, edge, or subgraph detection.

   **Definition 7.6.** *(Event Detection).*
   *Given* $\boldsymbol{G}$*, and a scoring function* $f : G_t \rightarrow \mathbb{R}$*, an event is defined as a time point* $t$*, such that* $|f(G_t) - f(G_{t-1})| > C_0$*, and* $|f(G_t) - f(G_{t+1})| > C_0$*.*

It is important to note the difference between event and change detection. While events represent isolated incidents, change points mark a point in time where the entire behavior of the graph changes and the difference is maintained until the next change point. One of the most popular applications of change detection is in networks modeling human interactions, such as communication and coauthorship networks.

**Definition 7.7.** *(Change Detection).*
*Given* $\boldsymbol{G}$*, and a scoring function* $f : G_t \to \mathbb{R}$*, an change is defined as a time point* $t$*, such that* $|f(G_t) - f(G_{t-1})| > C_0$*, and* $|f(G_t) - f(G_{t+1})| \leq C_0$*.*

### 7.5.2 Methods

In this section, some known methods for dynamic graph anomaly detection have been considered.

#### Community Based Anomaly Detection

In case of community-based methods, instead of monitoring the changes in entire network, a community is being monitored over time and an event is reported when there is a structural change. Chen *et al.* [20] proposed six types of community-based anomalies: shrink, grow, merge, split, born, and vanish. The authors developed a method based on graph representatives and community representatives to reduce the computational cost. Another type of community-based anomaly was proposed in [9] where the algorithm COM2, tracks comet communities over time. COM2, a novel and fast, incremental tensor analysis approach, which can discover both transient and periodic communities. This algorithm utilizes a novel Minimum Description Length (MDL) based formulation of the problem, that allows for parameter-free community search. The method is (a) scalable, being linear on the input size (b) general, (c) needs no user-defined parameters and (d) effective, returning results that agree with intuition.

#### Compression Based Anomaly Detection

In this methods a compact graph representation is achieved using MDL principle and compression techniques by exploiting patterns and regularity in the data with minimum encoding cost. Duan *et al.* [28] detect change-points in dynamic weighted directed graphs. For change-point detection, a measure of the similarity between partitions is presented to determine whether a change-point appears along the time axis. Sun *et al.* [88] proposed an algorithm called GraphScope, a parameter-free scheme to mine streams of graphs and monitoring their evolution over the time in order to detect abnormal events. GraphScope operates completely automatically, based on the Minimum Description

Length (MDL) principle. Furthermore, it adapts to the dynamic environment by automatically finding the communities and determining good change-points in time.

### Decomposition Based Anomaly Detection

These technique detects temporal anomalies by representing set of time evolving graphs as a matrix or tensor and performing factorization or dimensionality reduction. Sun *et al.* [90] proposed a novel method called Compact Matrix Decomposition (CMD) to compute sparse low rank approximation of a given matrix. The reconstruction values of each sparse graph is tracked over time and used for event detection [51].

### Distance Based Anomaly Detection

Utilizing the distance as a metric can be exploited to measure change between two input graphs. Two objects that have a small difference in measured metric are said to be similar. The metric measured in graphs are typically structural features, as the number of vertices [77]. There are various metrics for detecting anomalies: Maximum Common Subgraph (MCS), Error correcting graph matching distance, Graph Edit Distance (GED), Hamming distance for the adjacency matrices of the graphs, and so on. The methods that show better performances, are GED and MCS, however both distances are NP-hard. Gaston *et al.* [39] proposed a method to detect abnormal changes in time-evolving communication graphs using diameter distance, which is a measure of difference in graph diameter (specifically the greatest of the longest shortest paths for all vertices).

### Probabilistic Model Based Anomaly Detection

These methods, first construct a model which represents normality and, them flags deviations from this model as anomalous. The Bayesian anomaly detection method is presented in Heard *et al.* [49]. The authors focus on detecting anomalies in dynamic graphs using a two-stage method: the first stage uses simple, conjugate Bayesian models for discrete time counting processes to track the pairwise links of all nodes in the graph for assessing normal behavior; the second stage applies standard network inference on the greatly reduced subset of potentially anomalous nodes. In probabilistic methods, the anomaly detectors do not always perform a hard mapping from features to anomalies, but instead provides a likelihood that the structure is anomalous [77].

**Windows Based Anomaly Detection**

Anomaly detection algorithms provide some methods that are bound to a time window in order to spot anomalies. Mongiovi *et al.* [66] tackled the problem of detecting contiguous regions in graphs that are anomalous over time. Eberle *et al.* [29] proposed a novel approach called Pattern Learning and Anomaly Detection on Streams (PLADS), which is a partitioning and windowing approach that partitions the graph as it streams in over time and maintains a set of normative patterns and anomalies.

## 7.6 Outlier detection techniques

In the following section, we discuss the basic outlier detection techniques in static and dynamic graphs.

- Oddball. The aim of this technique, is to extract egonet-based features and patterns in order to spot anomalous nodes in weighted graphs [4]. An *egonet* is defined as the one-step neighborhood around a central node. OddBall, is a fast, unsupervised method to detect abnormal nodes in weighted graphs. This method does not require any user-defined constants. It also assigns an outlierness score to each node.
- Structural Clustering Algorithm for Networks. The purpose of this technique is to identify clusters, hubs and outliers in networks [104]. Two vertices are assigned to a cluster according to how they share neighbors. As such, vertices that are bridging many clusters are labeled as hubs, whereas those that are marginally connected to any community are flagged as outliers.
- Autopart – Parameter-Free Graph Partitioning. The authors develops parameter-free, iterative algorithms based on the Minimum Description Length principle for finding node groups and anomalous edges [17]. This technique specifically uses the adjacency matrix as graph representation, as well as for finding the best number of node groups automatically. AutoPart, an algorithm that automatically partitions the graph into clusters without user intervention, is capable of identifying anomalous edges.
- GOutRANK. The graph outlier ranking method (GOutRank) aims to detect anomalous nodes in attributed graphs. It generalizes outlier ranking method OutRank which has focused on high dimensional vector data without considering graph structures. Both techniques share the idea of computing a subspace clustering as pre-processing to the outlier ranking. GOutRank, detects outliers that can not be detected by traditional techniques. GOutRank is the first solution to outlier ranking in subspaces of attributed graphs [67].
- Deltacon. Koutra *et al.* [57] propose a principled, intuitive, and scalable algorithm that assesses the similarity between two graphs on the same

nodes. The similarity scores are calculated for two consecutive time steps, and as similarity between the two graphs is used the rooted Matusita Distance (which is related to the Euclidean Distance) between the score matrices. This technique uses belief propagation to measure network similarity.

- GOutlier. Aggarwal *et al.* [1] designed a method for outlier detection in graph streams with the use of a structural reservoir sampling approach for structural summarization. The method proposed is designed in order to create robust, dynamic and efficient models for outlier detection in graph streams.
- ECOutliers. The authors developed an iterative algorithm of detecting nodes which, over time, behave differently from the rest of community members. Such nodes are called Evolutionary Community Outliers [45]. ECOutlier, matching the time-evolving communities, and detecting the evolutionary community outliers.

## 7.7 Graph-based anomaly detection in real world applications

Anomaly detection in graphs is an area that has received much attention in recent years. Several techniques have been developed for anomaly detection in real-world. It has a wide variety of applications, including fraud and spam detection [5]. The authors highlight two main advantages of graph-based fraud detection techniques: 1) by word of mouth where the acquaintances of a fraudster can be considered as more likely to also commit fraud and 2) by collaboration where closely related parties come together to commit fraud. In both cases, the relational closeness can be exploited with graph-based detection techniques.

Anomaly detection techniques have widely been used in telecommunication networks [21], auction networks [19], accounting networks [64], security networks [68], opinion networks (e.g. deception and fake reviews) [32, 96, 3], financial trading networks [58], the web network (e.g. spam and malware) [16, 59, 15], social networks [35, 76], and computer networks (e.g. cyber-attacks and intrusion) [27].

In a real-world scenario, this approach would be applied to data such as cargo shipments, telecommunication traffic, financial transactions or terrorist networks. In all scenarios, the data consists of a series of nodes and links that share common nodes and links.

- Anomalies in telecommunication networks, one of the most prevalent is known as the subscription fraud. In this type of fraud, the fraudster often acquires an account using false identity with the intention of using the service for free.

- Anomalies in auction networks, the majority of online auction frauds occurs as non-delivery fraud, where the seller fails to deliver/ship the purchased goods to the buyer.
- Anomalies in accounting networks, SNARE (Social Network Analysis for Risk Evaluation) involves the task of spotting high-risk accounts with suspicious transactions behavior. Many existing techniques for detection rely on domain knowledge and rule-based signals e.g., based on large number of returns, many late postings, round-dollar entries, etc.
- Anomalies in security networks, relational learning has also been used in securities fraud detection where the task is to spot securities brokers that are likely to commit fraud and other violations of securities regulations in the future.
- Anomalies in the Web network, one of the main techniques in combating spam and malware on the Web has been to use trust and distrust propagation over the graph structure.
- Anomalies in social networks, another group of malware detection methods focuses on social malware in social networks such as Facebook. Such malware is also called socware. Socware consists of any posts appearing in ones news feed in social media platforms such as Facebook and Twitter that lead the user to malicious sites, make the user redistribute (e.g., by sharing/re-posting), and so on.
- Anomalies in computer networks, most graph-based network intrusion detection methods focus on the dynamically growing and changing nature of the network graph. In this graph, the nodes represent the agents in the networks, such as ad/file/directory servers and client nodes, and edges represent their communications over the network.

# 8

# Contributions

The problem tackled with here can be defined as follows:

**Anomaly detection in timed networks.** *Given a timed network, that is a network where each arc is equipped with a timestamp denoting the time of creation of the corresponding link, find the nodes in the network that are considerably dissimilar with respect to the rest of the network nodes when both the structure of their neighborhood and the order in which the structure has been established are taken into account.*

Different approaches have been proposed in the literature that search for anomalies in dynamic networks, among them [97, 54, 44, 66, 20].

We point out that the approach here proposed is substantially different from techniques dealing with dynamic networks. Indeed, our aim is not to determine the points in time in which a certain portion of the networks (typically a community or a subgraph) exhibited a significant change, as usually done by dynamic-graph anomaly detection techniques. Rather, our primary aim is to analyze each single node by taking simultaneously into account its temporal footprint.

In this sense our approach can be regarded as a static-graph anomaly detection technique in which temporal information has a privileged role in characterizing the behavior of network nodes.

This chapter is organized as follows. Section 8.1 reports preliminary notions and describe how the individual behavior is modeled; subsequent Section 8.2 illustrates the specific behavior models we retrieve to detect outliers; Section 8.3 is devoted to discuss the outlier score and its properties; Section 8.4 presents the several phases of the mining algorithm; Section 8.5 depicts the experiments we conduct on real datasets; finally, Section 9 concludes the work.

## 8.1 Behaviors on timed networks

In this section, we report the preliminary definitions and the notations employed throughout the paper. We aim at modeling the behavior of a node in the network through the way the node has interacted with its neighborhood during the time. Thus, first of all we introduce the model of network equipped with time information tackled by the proposed technique.

**Definition 8.1 (Timed Network).** *A* timed network *(or, simply,* network*) is a triple* $\mathcal{N} = (V, E, \tau)$*, where* $V = \{v_1, \ldots, v_n\}$ *is a set of* nodes*,* $E = \{e_1, \ldots, e_m\}$ *is a set of* arcs*, with each* $e_i = \langle s_i, d_i \rangle$ *an ordered pair of nodes in* $V$*, and* $\tau$ *a function associating each arc* $\langle s, d \rangle$ *in* $E$ *with a timestamp representing the instant of time in which the connection from* $s$ *to* $d$ *is established.*

Moreover, given a node $v$, we refer to the set of nodes $v'$ such that there is an arc from $v$ to $v'$ as the set of *outgoing neighbors* of $v$ (or, simply, neighbors) and we denote it as $\overrightarrow{N}(v)$. Vice versa, the set of nodes $v'$ such that there is an arc from $v'$ to $v$ is referred to as the set of *ingoing neighbors* of $v$ and we denote it as $\overleftarrow{N}(v)$. Finally, the total number of outgoing and ingoing neighbors is denoted as $\deg(v)$, then:

$$\deg(v) = |\overrightarrow{N}(v)| + |\overleftarrow{N}(v)|$$

Next, we provide formal definition of *contact* and *awareness* between nodes that are exploited for modeling interactions.

Given a network $\mathcal{N} = (V, E, \tau)$ and two nodes $v$ and $v'$in $V$, we say that $v$ *contacts* $v'$ at time $t$ if $\langle v, v' \rangle \in E$ and $\tau(\langle v, v' \rangle) = t$. Also, in this case, we say that $v'$ is a contact of $v$ starting from the instant of time $t$.

For example, in Figure 8.1a, $v$ contacts $v'$ at time $t = 10$ and, hence, starting from that time, $v'$ is a contact of $v$.

An *interaction* between two nodes $v$ and $v'$ is fired (or established) at time $t$ if either $v$ contacts $v'$ at time $t$ or $v'$ contacts $v$ at time $t$. In such a case, the established contact is said to be the *contact associated with the interaction.*

Given an interaction $i$ between two nodes $v$ and $v'$, the inverse interaction of $i$ is the establishment of the contact inverse with respect to the contact associated with $i$.

Thus, in Figure 8.1a and 8.1b, there is an interaction between $v$ and $v'$ fired at time $t = 10$ having as associated contact the arc from $v$ to $v'$ and, then, the inverse interaction between $v'$ and $v$ is fired at time $t = 20$, having as associated contact the arc from $v'$ to $v$.

Next we provide the definition of *awareness* which intends to model the intuition that an individual knows another individual, which is not one of its contacts, due to the presence of a common friend.

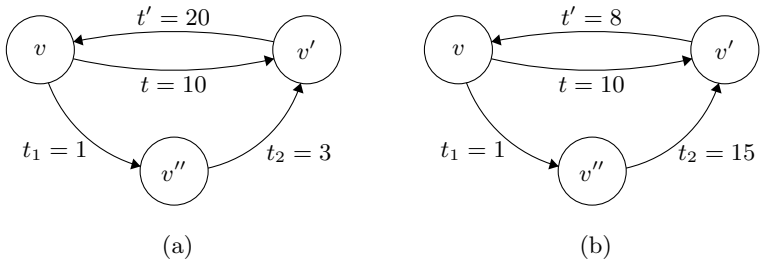



Fig. 8.1: Example

**Definition 8.2 (Awareness).** *Given a network $\mathcal{N} = (V, E, \tau)$ and two nodes $v$ and $v'$ in $V$, we say that the node $v$ is* mediately aware *(or, simply, aware) of $v'$ at time $t_a$ if there exists a node $v''$ in the network $\mathcal{N}$, such that $v$ contacts $v''$ at time $t_1$, $v''$ contacts $v'$ at time $t_2$, $\max\{t_1, t_2\} \leq t_a$ and (i) either $\langle v, v' \rangle$ is not in $E$ or (ii) $t_a \leq \tau(\langle v, v' \rangle)$. Moreover, we call* intermediary *the node $v''$ responsible of the awareness.*

Note that, according to our definition, $v$ is no more aware of $v'$ once $v'$ becomes a contact of $v$, since with the awareness we want to model the mediated knowledge between individuals.

In Figure 8.1a, $v$ is aware of $v'$ at each instant of time in the range $[3, 9]$. Starting from the instant of time $t = 10$, $v'$ becomes a contact of $v$ and, then, $v$ is no more aware of $v'$. Conversely, in Figure 8.1b, $v$ is never aware of $v'$ and, starting from the instant of time $t = 10$, $v'$ becomes a contact of $v$.

The notions of *contacts* and *awareness* are next exploited to model the behavior of a node within its neighborhood and, in particular, we distinguish between two families of behaviors: *action behavior* and *reaction behavior*.

**Definition 8.3 (Action behavior).** *Let $\mathcal{N}$ be a network, an* action *is an interaction between two nodes $v$ and $v'$ of the network fired before that the inverse interaction is fired.*

Let $v$ be a node of a network $\mathcal{N}$ and let $v'$ be one of its neighbor. According to the above definition, the action behaviors involving $v$ can be both the establishing of a connection from $v$ to $v'$ *preceding* a possibly connection from $v'$ to $v$ and the establishing of a connection from $v'$ to $v$ *preceding* a possibly connection from $v$ to $v'$.

Consider Figure 8.1a. There are two actions involving $v$: ($i$) the contact from $v$ to $v'$ which is accomplished before that $v'$ contacts $v$ and ($ii$) the contact from $v$ to $v''$. Consider, now, Figure 8.1b. There are again two actions involving $v$: ($i$) the contact from $v'$ to $v$ which is accomplished before that $v$ contacts $v'$ and ($ii$) the contact from $v$ to $v''$.

**Definition 8.4 (Reaction behavior).** *Let $\mathcal{N}$ be a network, a* reaction *is an interaction between two nodes $v$ and $v'$ of the network fired after that the inverse interaction is fired.*

The reaction behaviors involving $v$ are both the establishing of a connection from $v$ to $v'$ *succeeding* a connection from $v'$ to $v$ and the establishing of a connection from $v'$ to $v$ *succeeding* a connection from $v$ to $v'$.

Consider Figure 8.1a. There is one reaction involving $v$: the contact from $v'$ to $v$ which is accomplished after that $v'$ contacts $v$. Consider, now, Figure 8.1b. There is again one reaction involving $v$: the contact from $v'$ to $v$ which is accomplished after that $v$ contacts $v'$.

After having defined the concepts of actions and reactions, we can distinguish among different kinds of actions and reactions on the basis of the properties holding at the instant of time in which they are performed.

For example, Figures 8.1a and 8.1b depict two different kinds of actions:

*i.* the node ($v$) is aware of the other node ($v'$) when performs the action of contacting it (Figure 8.1a);
*ii.* the node ($v'$) is not aware of the other node ($v$) when performs the action of contacting it (Figure 8.1b).

In the following Section 8.2 we will describe in details which kinds of actions and reactions are addressed in this work.

The technique we propose aims at detecting outliers on the basis of their behavior taking simultaneously into account and suitably combining actions and reactions. Specifically, each action–reaction couple models a different *scenario* and we will denote it with the expression $A \leftrightarrow R$, where $A$ is the action and $R$ the reaction.

For example, consider the Twitter social network and consider the scenario in which an individual $v$ starts to follow another individual $v'$ and $v'$ does not follow back $v$. There, we can individuate an action performed by $v$ and received by $v'$ and a reaction performed by $v'$ (the decision of not following back $v$) and received by $v$.

For each scenario there are several involved performers: there is the performer who makes the action, the performer who receives the action, the performer who makes the reaction, the performer who receives the reaction and, in some cases, the performer involved as intermediary. Thus, on each scenario a node can play different roles. We call *structure* the coupling of role and scenario. Each structure defines a precise role played on a precise scenario and is referred to a single node called *actor* of the structure.

The previous example induces, then, four structures:

$s_1$: node making action (who decides on its own initiative to follow another node);
$s_2$: node receiving action (who is followed by another node on its own initiative);
$s_3$: node making reaction (who decides to do not follow back a node by which it was followed);
$s_4$: node receiving reaction (who is not followed back by a followed node).

For structures $s_1$ and $s_4$ the actor is $v$ while for the structures $s_2$ and $s_3$ the actor is $v'$.

Hence, once actions and reactions have been defined, those draw several scenarios and relative roles played. Scenarios and associated roles induce a set $\mathcal{S}$ of structures which encodes the node behavior. In particular, evaluating how frequently a node $v$ plays each possible role on each scenario (then, how frequent $v$ is the actor of each structure) leads to the building of the vector $\phi(v) = (\phi_{s_1}(v), \ldots, \phi_{s_k}(v))$ which represents the distribution of the roles played by the node on the different scenarios and $\phi_{s_i}(v)$ represents how frequently $v$ is the actor of the structure $s_i$. This distribution semantically encodes the *behavior* of the node in the network and can be effectively exploited to find anomalous individuals, as detailed in the Section 8.3.

## 8.2 Modeled behaviors

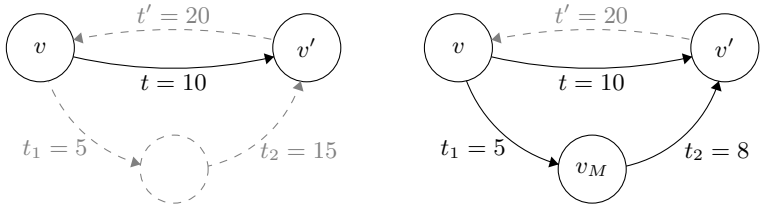
This section is devoted to present the behaviors considered. In particular, we present the kinds of actions and reactions we capture to gather information for modeling the overall node behavior. However, the approach is easily extensible to cover other kinds of actions/reactions.

Given a node $v$ and one of its neighbor $v'$, next we present the actions taken into account to model the behavior of $v$ and start by summarizing the notation employed:

$t$ the instant of time $\tau(\langle v, v' \rangle)$ associated with $\langle v, v' \rangle$;
$t'$ the instant of time $\tau(\langle v', v \rangle)$ if $\langle v', v \rangle \in E$; if $\langle v', v \rangle$ is not in $E$ then $t'$ is set to $-1$, meaning that it is not defined;
$t_M$ the greatest instant of time smaller than $t$ such that $v$ is aware of $v'$ at time $t_M$ due to the intermediary $v''$, and we refer as $v_M$ the node $v''$; if $v$ was not aware of $v'$ when the connection from $v$ and $v'$ was established then $t_M$ is set to $-1$, meaning that it is not defined.

For the sake of readability, we employ $\texttt{def}(t)$ for indicating that $t \neq -1$ and $\texttt{undef}(t)$ for indicating that $t = -1$.

For each considered action/reaction, we discuss the semantic behavior associated with it together with the conditions to be checked in order to verify if the behavior under analysis is actually assumed.




(a) Action 1 (b) Action 2

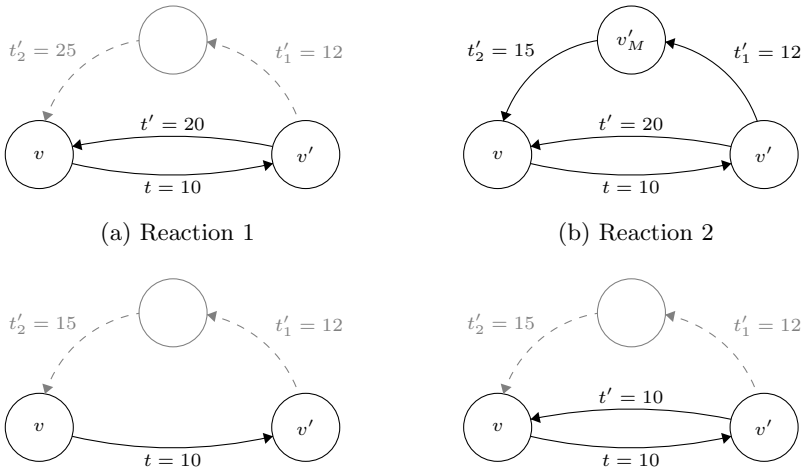
Fig. 8.2: Action behaviors

*(Action 1)* **a node contacts another node on its own initiative**.
This action represents that $v$ contacts $v'$ before that $v'$ contacts $v$ and without $v$ being aware of $v'$ (see Fig. 8.2a).

***Condition:*** $(\texttt{undef}(t') \vee t \leq t') \bigwedge \texttt{undef}(t_M)$

*(Action 2)* **a node contacts another node due to an intermediary**.
This structure means that $v$ contacts $v'$ before that $v'$ contacts $v$ but after that $v$ becomes aware of $v'$ (see Fig. 8.2b).

***Condition:*** $(\texttt{undef}(t') \vee t \leq t') \bigwedge \texttt{def}(t_M) \bigwedge t_M < t$




(a) Reaction 1 (b) Reaction 2

(c) Reaction 3 (d) Reaction 4

Fig. 8.3: Reaction behaviors

As for the reactions, we capture four kinds of reactions for $v$. Note that, since these are reactions, we assume that either a connection from $v$ to $v'$ or a connection from $v'$ to $v$ has already been fired.

*(Reaction 1)* **a node directly replies to the node who contacts him**:
This reaction models that $v'$ contacts $v$ since $v$ contacts $v'$ and not because $v'$ becomes aware of $v$ (see Fig. 8.3a).

***Condition:*** $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge t < t' \bigwedge (\texttt{undef}(t'_M) \vee t'_M < t)$

*(Reaction 2)* **a node replies to the node who contacts him due to an intermediary**:
This reaction represents that $v'$ contacts $v$ after that $v$ contacts $v'$ but only after that $v'$ becomes aware of $v$ (see Fig. 8.3b).

***Condition:*** $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge \texttt{def}(t'_M) \bigwedge t < t'_M < t'$

*(Reaction 3)* **a node does not reply to the node who contacts him**:
This is not an actual reaction since it represents that $v'$ does not react to the action performed by $v$ (see Fig. 8.3c).

***Condition:*** $\texttt{def}(t) \bigwedge \texttt{undef}(t')$

*(Reaction 4)* **a node contacts the node who contacts him independently**:
This is not an actual reaction since it represents that $v'$ contacts $v$ on its own initiative (see Fig. 8.3d).

***Condition:*** $\texttt{def}(t) \bigwedge \texttt{def}(t') \bigwedge t' = t \bigwedge \texttt{undef}(t'_M)$

Once actions and reactions are defined, we can analyze which scenarios are modeled and which structures are induced. In particular, we have eight different scenarios and for each scenario two roles are definable, the node who acts and the node who reacts. Moreover, for scenarios involving action $A_2$ and/or reaction $R_2$, also the role of intermediary is definable.

Thus, focusing on a single node $v$, we can define seventeen structures for it:

structures $s_1 \dots s_4$: $v$ plays the role of performing action $A_1$ and receiving one of the possible four reactions;
structures $s_5 \dots s_8$: $v$ plays the role of performing action $A_2$ and receiving one of the possible four reactions;
structures $s_9 \dots s_{12}$: $v$ plays the role of receiving action $A_1$ and performing one of the possible four reactions;
structures $s_{13} \dots s_{16}$: $v$ plays the role of receiving action $A_2$ and performing one of the possible four reactions;
structure $s_{17}$: $v$ plays the role of being the intermediary of a couple of interacting nodes.

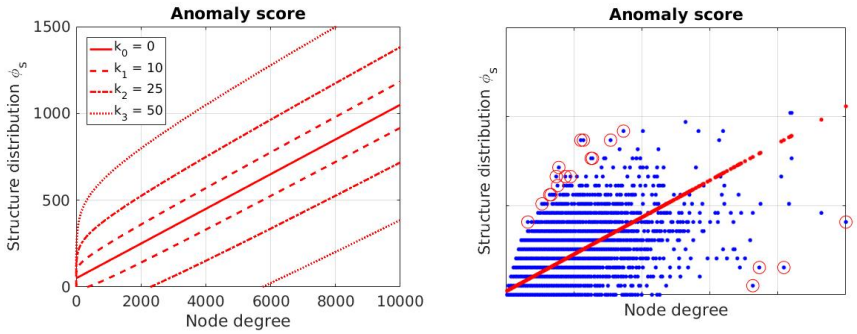



Fig. 8.4: Example of bandwidths associated with the score of a structure (left) and example highlighting the top twenty anomalies on a real dataset (right).

For example, consider the scenario depicted in Figure 8.1a again: *$v$ contacts $v'$ after viewing that $v'$ is a contact of $v''$ which is one of its contact. $v'$ reacts to this contacting back $v$*. The scenario is then $A_2 \leftrightarrow R_1$ and the roles are: (1) who makes the action $A_2$ and receives the action $R_1$, (2) who receives the action $A_2$ and makes the reaction $R_1$, and (3) who plays the role of intermediary. Thus, after analyzing this scenario, for the node $v$ we have to update the structure associated with (1), that is $s_5$; for the node $v'$ the structure associated with (2), that is $s_{13}$; and for the node $v''$ the structure associated with (3), that is $s_{17}$.

## 8.3 Anomaly Score

The distribution $\phi_s(v)$ encodes the behavior of the node $v$ in terms of how much frequently it is involved in the different structures. We can then exploit the distributions $\phi_s$ in order to determine how typical is the behavior of each node with respect to the whole population. With this aim, an anomaly score is assigned to each node.

Given the network $\mathcal{N} = (V, E, \tau)$, for each structure $s \in \mathcal{S}$ the regression line of the set of points $P_s(\mathcal{N}) = \{(\deg(v_i), \phi_s(v_i)) \mid v_i \in V\}$ is computed.

Let $\alpha_s$ and $\beta_s$ denote be the parameters of the estimated line. The anomaly score of the node $v_i$ with respect the structure $s$ is defined as:

$$\mathrm{sc}_s(v_i) = \left| \frac{\phi_s(v_i) - [\alpha_s \cdot \deg(v_i) + \beta_s]}{\log_2(1 + \deg(v_i))} \right| \tag{8.1}$$

The numerator of Equation (8.1) represents the deviation of the observed number of structures $\phi_s(v_i)$ from the expected one $y_i$, according to the value predicted by the regression curve $y_i = \alpha_s \cdot \deg(v_i) + \beta_s$. As for the denominator, it serves the purpose of taking into account the cardinality of the neighborhood of $v_i$, while the absolute value is needed to capture both the upper tail and the lower tail of resulting distribution.

Figure 8.4 on the left reports a regression line (the solid curve for $k_0 = 0$, having parameters $\alpha = 0.1$ and $\beta = 50$) and the bandwidths associated with different values of anomaly score (specifically, for $k_1 = 10$, $k_2 = 25$, and $k_3 = 50$; e.g., the score associated with points falling within the dashed bandwidth will be not greater than $k_1$).

Figure 8.4 on the right reports the structure distribution associated with a real dataset and the top twenty anomalies according to Equation (8.1).

Scores associated with each single structure are then normalized in order to make them homogeneous

$$\widehat{\mathrm{sc}}_s(v_i) = \frac{\mathrm{sc}_s(v_i)}{\mathrm{std}(\{\mathrm{sc}_s(\mathbf{V})\})} \tag{8.2}$$

and, hence, expressed in terms of number of standard deviations.

The anomaly score of a node $v_i$ is

$$\mathrm{sc}(x_i) = \sum_s \widehat{\mathrm{sc}}_s(x_i), \tag{8.3}$$

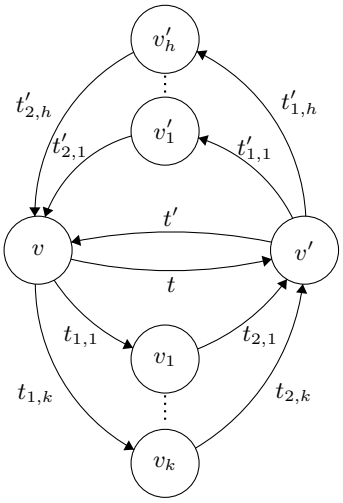
that is obtained by combining the scores computed with respect to the single structures.

## 8.4 Algorithm

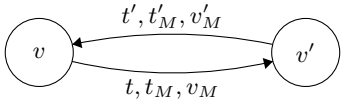
In this section the algorithm we designed to mine outliers is presented and its properties are discussed. The algorithm consists in three main phases each accounted next.

**Phase 1.** This phase has the intent of enriching the information associated with the arcs (see Figure 8.5). In order to retrieve behaviors illustrated in Section 8.2 we need to find, for each arc $\langle v, v' \rangle$ with associated timestamp $t$, if $v$ is aware of $v'$ at time $t$, namely we have to search for a node $\hat{v}$ such that both edge $e_1 = \langle v, \hat{v} \rangle$ and edge $e_2 = \langle v, \hat{v} \rangle$ exist and the timestamps $t_1$ and $t_2$ associated with these edges are both strictly smaller than $t$. Among these nodes, we are interested in the node $v_M$ which is the most recent responsible of the fact that $v$ is aware of $v'$. Finally, the edge $e = \langle v, v' \rangle$ is annotated with the node $v_M^e$ and the time $t_M^e$ which is the instant of time starting by which $v$ is aware of $v'$ due to $v_M^e$; in formula $t_M^e$ is the maximum between the time associated with the arc $\langle v, v_M^e \rangle$ and the time associated with the arc $\langle v_M^e, v' \rangle$. Concluding, given a network $\mathcal{N} = (V, E, \tau)$, the phase returns the annotated network $\mathcal{N}^+ = (V, E, \tau^+)$ where $\tau^+(e)$ returns the triple $(\tau(e), t_M^e, v_M^e)$.

**Computational complexity of Phase 1.** As for the cost of this phase, let $\mathcal{N} = (V, E, \tau)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. We iterate over the set of edges and for each arc $e = \langle v, v' \rangle$ in $E$ we iterate over the set $\overrightarrow{N}(v)$ of neighbors of $v$ in order to search $v_M$ and, then, for each neighbor $\hat{v}$ of $v$ we have to check if there exists an arc from it to $v'$. This




(a) analyzed pattern




where $t_M = \max\{t_i = \max\{t_{1,i}, t_{2,i}\} \mid t_i < t\}$
$v_M = v_i$ s. t. $\max\{t_{1,i}, t_{2,i}\} = t_M$
$t'_M = \max\{t'_i = \max\{t'_{1,i}, t'_{2,i}\} \mid t'_i < t'\}$
$v'_M = v'_i$ s. t. $\max\{t'_{1,i}, t'_{2,i}\} = t'_M$

(b) annotated graph

Fig. 8.5: Graph annotation (Phase 1)

latter operation can be performed through a binary search in the list of the outgoing arcs of $\hat{v}$. Thus, the overall cost is

$$\sum_{\langle v,v' \rangle} \sum_{\hat{v} \in \overrightarrow{N}(v)} \log \overrightarrow{N}(\hat{v}) = O(m \cdot \overline{n} \cdot \log \overline{n}) \tag{8.4}$$

where $\overline{n}$ denotes the maximum number of neighbors of a node generic in the networks.

**Phase 2.** This phase has the intent of mining the behavior of each individual in the network, starting from the annotated network coming from the previous phase. Then, given an annotated network $\mathcal{N}^+ = (V, E, \tau^+)$ we iterate over the set of nodes $V$ and for each node $v$ in $V$ we iterate over the set of neighbors and for each neighbor $v'$ in $\overrightarrow{N}(v)$ the behaviors depicted in Section 8.2 are evaluated. In particular, through the information provided by $\mathcal{N}^+$ the conditions associated with the behaviors are checked and, according to the result of the check, the behavior counters are updated. The result of this phase is then the distribution of the behaviors for each node.

**Computational complexity of Phase 2.** As for the cost of this phase, let $\mathcal{N}^+ = (V, E, \tau^+)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. Iterating over the set of nodes and for each node $v$ iterating over the set of neighbors $\overrightarrow{N}(v)$ corresponds to iterating over the set of edges. For each edge, in constant time we can obtain the required information by $\mathcal{N}^+$ and we can evaluate all the conditions. Since the number of conditions is fixed, also this

**Phase 1:** Network information enrichment

**Input:** A network $\mathcal{N} = (V, E, \tau)$
**Output:** The annotated network $\mathcal{N}^+$

**foreach** edge $e = \langle v, v' \rangle$ in $E$ **do**
  let $t = \tau(e)$ be the timestamp associated with the edge from $v$ to $v'$;
  set $t_M$ to $-1$;
  set $v_M$ to $\emptyset$;
  **foreach** edge $\hat{e} = \langle v, \hat{v} \rangle$ **do**
    let $t_1 = \tau(\hat{e})$ be the timestamp associated with the edge from $v$ to $\hat{v}$;
    **if** $\langle \hat{v}, v' \rangle$ belongs to $E$ **then**
      let $t_2 = \tau(\langle \hat{v}, v' \rangle)$ be the timestamp associated with the edge from $\hat{v}$ to $v'$;
      let $\hat{t}$ be $\max\{t_1, t_2\}$;
      **if** $t_M < \hat{t} < t$ **then**
        set $t_M$ to $\hat{t}$;
        set $v_M$ to $\hat{v}$;
  associate $t_M$ and $v_M$ with the $e$;
  `// then substitute` $\tau(e) = t$ `with` $\tau^+(e) = (t, t_M, v_M)$– see Fig. 8.5

latter step can be accomplished in constant time. Thus, the overall cost of this phase is $O(m)$.

**Phase 2:** Structures computation

**Input:** An annotated network $\mathcal{N}^+ = (V, E, \tau^+)$
**Output:** The distribution of structures $\phi_v$ for each node $v$

**foreach** node $v$ in $V$ **do**
  set $\phi_s(v)$ to 0 for each structure $s$;
  **foreach** neighbor $v'$ of $v$ **do**
    extract tuple $T = \big(\tau^+(\langle v, v' \rangle), \tau^+(\langle v', v \rangle)\big)$;
    `// T contains then` $(t, t_M, v_M, t', t'_M, v'_M)$– see Fig. 8.5b
    **foreach** action $a$ **do**
      **foreach** reaction $r$ **do**
        let $s$ be the structure associated with the pair $a \leftrightarrow r$;
        **if** $C_a(T)$ **and** $C_r(T)$ **then**
          update $\phi_s(v)$;
        let $\hat{s}$ be the structure in which a node plays the role of being the intermediary of a couple of interaction nodes ($s_{17}$);
        **if** $a \leftrightarrow r$ involves node $v_M$ **then**
          update $\phi_{\hat{s}}(v_M)$;

**Phase 3.** This phase has the intent of detecting outlier individuals. Starting from the distribution of behaviors computed by the previous phase, we have to compute the outlier score as defined in Section 8.3. The first step consists in computing, for each considered structure, the regression line. The second step consists in computing for each structure $s$ and for each node $v$ the score $\mathrm{sc}_s(v)$ achieved by node $v$ on the structure $s$ by means of Equation (8.1). Next the standard deviation of the scores assumed by nodes on structure $s$ is computed and, then, this value is exploited to normalize the scores (lines 6–7).

After that all the structures have been analyzed, the outlier score of each node $v$ is computed by properly aggregating the score achieved by $v$ on each single structure by means of Equation (8.3).

**Computational complexity of Phase 3.** As for the cost of this phase, let $\mathcal{N}^+ = (V, E, \tau^+)$ be the analyzed network, let $n = |V|$ and let $m = |E|$. Computing the regression line has a cost linear with respect to the number $n$ of nodes. Next, for each node we had to compute the score. Since Equation (8.1) is computable in constant time, also this step has a cost linearly dependent from $n$. Normalizing the scores costs $O(n)$ as well and, finally, also computing the overall outlier score costs $O(n)$ since Equation (8.3) iterates over a fixed number of structures.

**Phase 3:** Outlier mining

**Input:** The distribution of structures $\phi_v$ for each node $v$
**Output:** The overall outlier score for each node $v$

```
1 foreach structure s do
2     compute the regression line of the observations (deg(v), φ_s(v));
3     foreach node v in N do
4         compute the score sc_s(V) of v for structure s through Eq. (8.1);
5     compute the standard deviation of the score std({sc_s(V)});
6     foreach node v in N do
7         compute the normalized score of v for structure s through Eq. (8.2);
8 foreach node v in N do
9     compute the overall score of v through Eq. (8.3);
```

## 8.5 Experimental results

In this section experimental results concerning the introduced technique are presented. All the datasets employed are from the *Online Social Networks Research*. All the dataset underwent a preprocessing during which multiple

and self links have been removed. The *Digg friends* dataset[1] contains data about stories promoted to Digg's front page[2] over a period of a month in 2009. The dataset contains Digg users who have voted for a story. We considered the voters' friendship links, where a link `user_id` → `friend_id` means that `user_id` is watching the activities of (is a fan of) `friend_id`. User identifiers are available already anonymized. The network analyzed consists of 279,631 nodes and 2,251,166 arcs. The *Facebook wall* dataset[3] contains a list of all of the wall posts from the Facebook New Orleans network. Each line contains two anonymized user identifiers, meaning the second user posted on the first user's wall. The third column is the times of the wall post. The network analyzed consists of 45,813 nodes and 264,004 arcs. The *Wikipedia growth*[4] dataset contains links between Wikipedia pages and the time when these links were first created. The dataset represents the complete history of the network over a period of 826 days, between January 1st, 2005 and April 6th, 2007. The datasets is anonymized to protect the privacy of page authors. The network analyzed consists of 1,870,709 nodes and 39,953,145 arcs.

The following table reports the total number of the structures mined, for each of the structures $s_i$ in the set $\mathcal{S}$. Notice that the total counts associated with the pairs of structures $(s_i, s_{i+8})$ are identical, since these structures are induced by symmetric roles in the same scenario.

| Structure | *Digg friends* | *Facebook wall* | *Wikipedia growth* |
|---|---|---|---|
| $s_1, s_9$ | 61,122 | 57,476 | 2,030,550 |
| $s_2, s_{10}$ | 8,307 | 4,167 | 641,677 |
| $s_3, s_{11}$ | 683,282 | 74,268 | 22,416,947 |
| $s_4, s_{12}$ | 6,294 | 0 | 7,596 |
| $s_5, s_{13}$ | 65,844 | 16,319 | 451,128 |
| $s_6, s_{14}$ | 44,301 | 2,629 | 293,043 |
| $s_7, s_{15}$ | 681,317 | 28,552 | 10,694,970 |
| $s_8, s_{16}$ | 152 | 0 | 56 |
| $s_{17}$ | 1,009,571 | 80,042 | 14,052,936 |
| *Total* | 7,574,125 | 446,864 | 87,124,870 |

Table 8.1: Total Number of the Structures Mined.

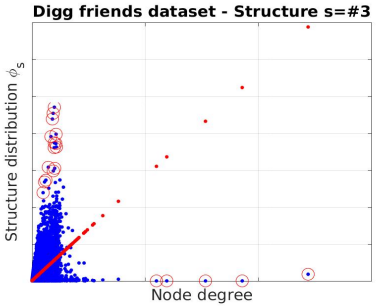
Clearly, this does not mean that they are redundant, since the respective counts per node differ in general, being different the number of times in which

[1] `http://www.isi.edu/integration/people/lerman/downloads.html`

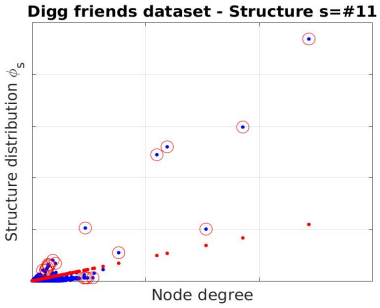
[2] *Digg* is a news aggregator (`http://digg.com`) aiming to select stories for the Internet audience such as science, trending political issues, and viral Internet issues. It allows people to vote web content up or down.

[3] `http://socialnetworks.mpi-sws.org/data-wosn2009.html`

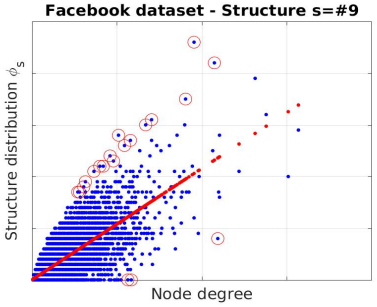
[4] `http://socialnetworks.mpi-sws.org/data-wosn2008.html`



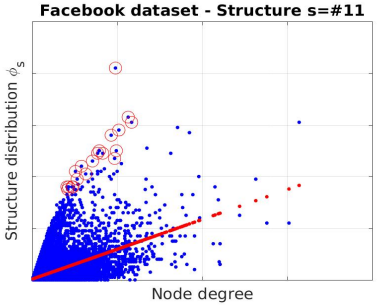

(a) Digg dataset – $s_3$



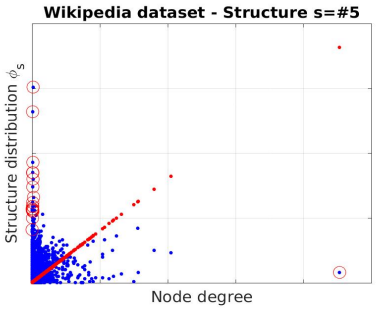

(b) Digg dataset – $s_{11}$



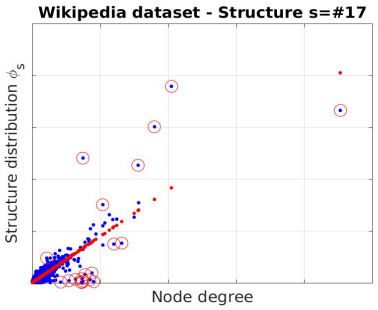

(c) Facebook dataset – $s_9$




(d) Facebook dataset – $s_{11}$




(e) Wikipedia growth dataset – $s_5$




(f) Wikipedia growth dataset – $s_{17}$

Fig. 8.6: Structure count distribution.

the single node plays each role in the same scenario. In the *Facebook wall* dataset the structures $s_4$, $s_8$, $s_{12}$, and $s_{16}$, capturing the simultaneity of the response, are not present since timestamps are almost all different (only 846 timestamps appear more than once in the dataset) and it is never the case that two users simultaneously make a post on the respective wall. In general,

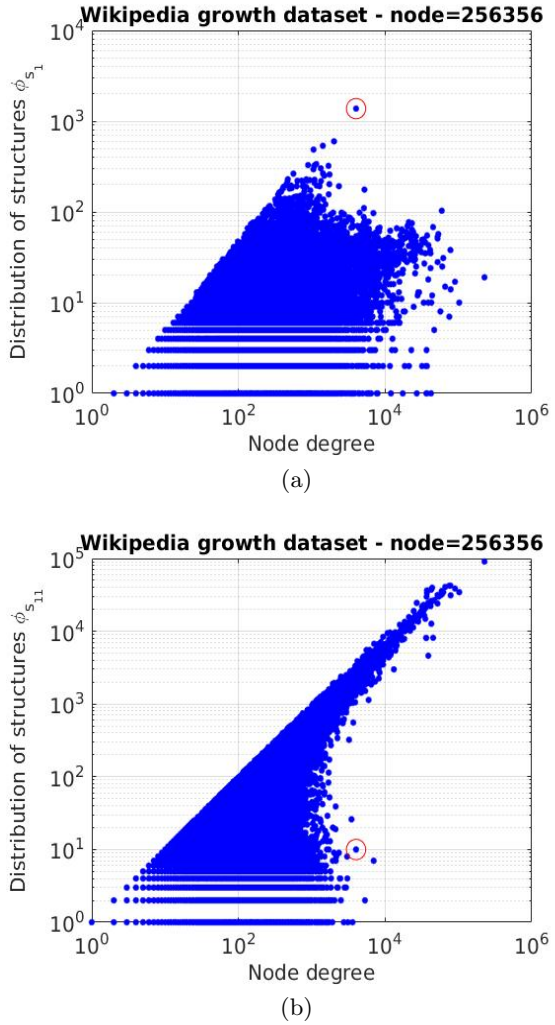



Fig. 8.7: Notable structure distributions for a top outlier node

these structures are among the less numerous in these datasets due to the fine granularity of the temporal scale.

Figure 8.6 shows the scatter plots of the node degree versus the structure count for some of the structures mined. Each plot reports also the regression line of the points (represented by the red points) and highlights the top 20 anomalies (the red circled points) according to the anomaly score of Equation (8.1).

Next, we discuss on the knowledge mined for one of the top outlier in a dataset in order to highlight how the proposed technique is able to provide

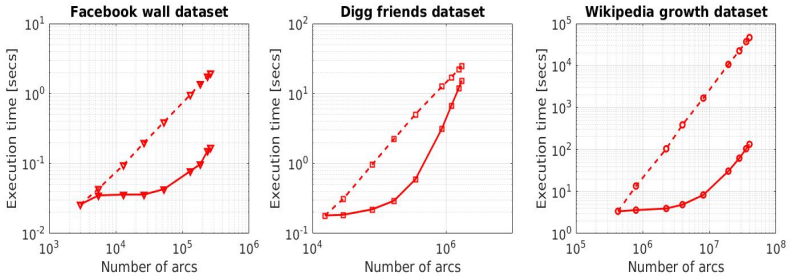



Fig. 8.8: Scalability analysis.

not only the outlier score but also an in intelligible interpretation of the score, shedding light on semantic underlying the decision made by the technique of signaling a node as anomalous.

Specifically, we focus on the *Wikipedia growth* dataset and on the node *out* having id $256,356$. We consider the two structures that mostly contribute to the large score achieved by *out*. Figure 8.7 reports how the number of times in which *out* is actor of structure $s_1$ and $s_{11}$ places the node with respect to the number of times in which the other nodes perform as actors of those structures.

It is clear by the plots that *out* is located in both cases at the margin of the distribution. In particular, *out* performs as actor of structure $s_1$ much more times than other nodes, while performs as actor of structure $s_{11}$ much less times than nodes having a similar neighborhood cardinality.

From a semantic point of view, these plots naturally lead to a description of the outlierness that could explain the exceptionality of the node. Since *out* very often plays as actor for structure $s_1$, the associated Wikipedia page has a high number of links and, exceptionally, almost always the linked page links back the page associated with *out*. Moreover, since *out* rarely plays as actor for structure $s_{11}$, whenever *out* is linked by a page $p$, rarely a link to $p$ does not appear in *out*.

Finally, Figure 8.8 shows the scalability of the method. We varied the size of the datasets, from the 1% to the 100% of the original data, by randomly sampling nodes and retaining the arcs linking only pairs of nodes in the selected sample. The solid lines in the plot show the total execution time versus the number of arcs of the network. The dashed lines represents the cost of the method as reported in Equation (8.4), by using as $\overline{n}$ the mean number of neighbors of nodes in the network, with a constant prefactor computed in a way such that the two curves start from the same point. The curves show that the actual cost of the method is generally below that predicted by the cost analysis, thus confirming the applicability of the method to large networks. To illustrate, the full *Wikipedia growth* dataset was processed in about 120 seconds on a Intel Core i7 2.40GHz equipped machine under the Linux operating system.

# 9

# Conclusions

This thesis investigated data mining techniques for large and complex networks. Two families of data mining algorithms have been considered, namely Classification and Anomaly Detection.

As for the classification task the focus was on large data. Motivated by approaches designed to improve generalization and to prevent induction of overly complex models, in this thesis we investigated the application of the Pessimistic Error Estimate (PEE) principle in the context of nearest neighbor rule competence preservation techniques. As major results, we showed that PEE-like selection strategies guarantee to preserve the accuracy of a nearest neighbor consistent subset with a far larger reduction factor and that sensible generalization improvements can be obtained by using a reduced subset of intermediate size. We have proposed four novel strategies based FCNN algorithm which is efficient and has low quadratic complexity. Our work is focused on reducing the runtime and preserve the accuracy. We have also compared our strategies with the competence enhancement of preservation state-of-the-art algorithms on real datasets.

As for the anomaly detection the focus was on complex network data. We considered the anomaly detection in timed networks problem whose goal is to single out anomalies by taking into account simultaneously information concerning both the structure of the network and the order in which connections have been established. Our primary aim is to analyzing each single node by taking simultaneously into account its temporal footprint. We defined a set of spatio-temporal structures is induced by checking certain conditions on the order of arc appearance denoting different kinds of user behaviors and exploited their distribution to detect anomalies. We presented a scalable algorithm and experimental results showing the peculiarity of the knowledge mined by our technique and its applicability to the analysis of large networks.

# References


1. Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. Outlier detection in graph streams. In *2011 IEEE 27th International Conference on Data Engineering*, pages 399–409. IEEE, 2011.
2. David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.
3. Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11, 2013.
4. Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.
5. Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.
6. Andry Alamsyah, Budi Rahardjo, et al. Community detection methods in social network analysis. *Advanced Science Letters*, 20(1):250–253, 2014.
7. Fabrizio Angiulli. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd international conference on Machine learning*, pages 25–32. ACM, 2005.
8. Fabrizio Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.
9. Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. Com2: fast automatic discovery of temporal (comet) communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.
10. Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.
11. Jeffrey P Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, pages 131–136. Springer, 1998.
12. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

13. Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.
14. Carla E Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 17–24, 1993.
15. Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th international conference on World wide web*, pages 197–206. ACM, 2011.
16. Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430. ACM, 2007.
17. Deepayan Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 112–124. Springer, 2004.
18. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.
19. Duen Horng Chau, Shashank Pandit, and Christos Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 103–114. Springer, 2006.
20. Zhengzhang Chen, William Hendrix, and Nagiza F Samatova. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85, 2012.
21. Corinna Cortes, Daryl Pregibon, and Chris Volinsky. Communities of interest. In *International Symposium on Intelligent Data Analysis*, pages 105–114. Springer, 2001.
22. Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.
23. Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.
24. V Susheela Devi and M Narasimha Murty. An incremental prototype set building technique. *Pattern Recognition*, 35(2):505–513, 2002.
25. Luc Devroye. On the inequality of cover and hart in nearest neighbor discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):75–78, 1981.
26. Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.
27. Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894. ACM, 2012.
28. Dongsheng Duan, Yuhua Li, Yanan Jin, and Zhengding Lu. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18. ACM, 2009.

29. William Eberle and Lawrence Holder. A partitioning approach to scaling anomaly detection in graph streams. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 17–24. IEEE, 2014.
30. Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):476–491, 1997.
31. Susana Eyheramendy, Alexander Genkin, Wen-Hua Ju, David D Lewis, and David Madigan. Sparse bayesian classifiers for text categorization. *Journal of Intelligence Community Research and Development*, 13, 2003.
32. Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. Distributional footprints of deceptive product reviews. *ICWSM*, 12:98–105, 2012.
33. Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.
34. Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.
35. Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. Towards online spam filtering in social networks. In *NDSS*, volume 12, pages 1–16, 2012.
36. Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 813–822. ACM, 2010.
37. Salvador García, José Ramón Cano, and Francisco Herrera. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, 41(8):2693–2709, 2008.
38. Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):417–435, 2012.
39. Matthew E Gaston, Miro Kraetzl, and Walter D Wallis. Using graph diameter for change detection in dynamic networks. *Australasian Journal of Combinatorics*, 35:299, 2006.
40. W. Gates. The reduced nearest neighbor rule. 1972.
41. M Gegick. Symmetric nonnegative matrix factorization for graph clustering. 2012.
42. Marion Gibbon and Durga Pokhrel. Social network analysis, social capital and their policy implications. *PLA Notes*, 36:29–33, 1999.
43. Peter Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004.
44. Manish Gupta, Jing Gao, Yizhou Sun, and Jiawei Han. Community trend outlier detection using soft temporal pattern mining. In *Machine Learning and Knowledge Discovery in Databases*, pages 692–708. Springer, 2012.
45. Manish Gupta, Jing Gao, Yizhou Sun, and Jiawei Han. Integrating community matching and outlier detection for mining evolutionary community outliers. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 859–867. ACM, 2012.
46. Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Elsevier, 2011.
47. Peter.E. Hart. The condensed nearest neighbor rule. 1968.

48. Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.
49. Nicholas A Heard, David J Weston, Kiriaki Platanioti, David J Hand, et al. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662, 2010.
50. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.
51. Tsuyoshi Idé and Hisashi Kashima. Eigenspace-based anomaly detection in computer systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 440–449. ACM, 2004.
52. Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.
53. David D Jensen and Paul R Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.
54. Tengfei Ji, Dongqing Yang, and Jun Gao. Incremental local evolutionary outlier detection for dynamic social networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 1–15. Springer, 2013.
55. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.
56. Bilge Karacali and Hamid Krim. Fast minimization of structural risk by nearest neighbor rule. *IEEE transactions on neural networks*, 14(1):127–137, 2003.
57. Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. D elta c on: A principled massive-graph similarity function. In *Proceedings of the SIAM International Conference in Data Mining. Society for Industrial and Applied Mathematics*, pages 162–170. SIAM, 2013.
58. Zhongmou Li, Hui Xiong, Yanchi Liu, and Aoying Zhou. Detecting blackhole and volcano patterns in directed networks. In *2010 IEEE International Conference on Data Mining*, pages 294–303. IEEE, 2010.
59. Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.
60. Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 195–201. Citeseer, 1997.
61. Elena Marchiori. Class conditional nearest neighbor for large margin instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):364–370, 2010.
62. Alexandra Marin and Barry Wellman. Social network analysis: An introduction. *The SAGE handbook of social network analysis*, pages 11–25, 2011.
63. Denny Matthew. Introduction to social network theory. *UMass ISSR Workshop*, 2014.
64. Mary McGlohon, Stephen Bay, Markus G Anderle, David M Steier, and Christos Faloutsos. Snare: a link analytic system for graph labeling and risk detection. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1265–1274. ACM, 2009.
65. John Mingers. Expert systemsrule induction with statistical data. *Journal of the operational research society*, 38(1):39–47, 1987.

66. Misael Mongiovi, Petko Bogdanov, Razvan Ranca, Ambuj K Singh, Evangelos E Papalexakis, and Christos Faloutsos. Netspot: Spotting significant anomalous regions on dynamic networks. In *Proceedings of the 13th SIAM international conference on data mining (SDM), Texas-Austin, TX*. SIAM, 2013.
67. Emmanuel Müller, Patricia Iglesias Sánchez, Yvonne Mülle, and Klemens Böhm. Ranking outlier nodes in subspaces of attributed graphs. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*, pages 216–222. IEEE, 2013.
68. Jennifer Neville, Özgür Şimşek, David Jensen, John Komoroske, Kelly Palmer, and Henry Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 449–458. ACM, 2005.
69. Tim Niblett and Ivan Bratko. Learning decision rules in noisy domains. In *Proceedings of Expert Systems' 86, The 6Th Annual Technical Conference on Research and development in expert systems III*, pages 25–34. Cambridge University Press, 1987.
70. Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.
71. Evelien Otte and Ronald Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, 28(6):441–453, 2002.
72. Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1346–1355. ACM, 2014.
73. J Ross Quinlan. Simplifying decision trees. *International Journal of Human-Computer Studies*, 51(2):497–510, 1999.
74. J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
75. J Ross Quinlan and Ronald L Rivest. Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248, 1989.
76. Md Sazzadur Rahman, Ting-Kai Huang, Harsha V Madhyastha, and Michalis Faloutsos. Efficient and scalable socware detection in online social networks. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 663–678, 2012.
77. Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.
78. Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *Journal of Machine Learning Research*, 15(1):335–366, 2014.
79. Jorma Rissanen. *Minimum description length principle*. Wiley Online Library, 1985.
80. GL Ritter, HB Woodruff, SR Lowry, and TL Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.
81. Cullen Schaffer. Overfitting avoidance as bias. *Machine learning*, 10(2):153–178, 1993.

82. John Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988.
83. John Scott. *Social network analysis*. Sage, 2012.
84. Olivier Serrat. Social network analysis. 2009.
85. David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the eleventh international conference on machine learning*, pages 293–301, 1994.
86. Charles J Stone. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977.
87. Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.
88. Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.
89. Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.
90. Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Statistical Analysis and Data Mining*, 1(1):6–22, 2008.
91. Pan-Ning Tan, Michael Steinbach, and Vipin Kumar. Classification: basic concepts, decision trees, and model evaluation. *Introduction to data mining*, 1:145–205, 2006.
92. Lei Tang and Huan Liu. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–137, 2010.
93. Ivan Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, (6):448–452, 1976.
94. Hanghang Tong and Ching-Yung Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SDM*, pages 143–153. SIAM, 2011.
95. Renee C Van der Hulst. Introduction to social network analysis (sna) as an investigative tool. *Trends in Organized Crime*, 12(2):101–121, 2009.
96. Guan Wang, Sihong Xie, Bing Liu, and Philip S Yu. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):61, 2012.
97. Teng Wang, Chunsheng Victor Fang, Derek Lin, and S Felix Wu. Localizing temporal anomalies in large evolving graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 927–935. SIAM.
98. Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
99. Charles Wetherell, Andrejs Plakans, and Barry Wellman. Social networks, kinship, and community in eastern europe. *The Journal of Interdisciplinary History*, 24(4):639–663, 1994.
100. D Randall Wilson and Tony R Martinez. Instance pruning techniques. In *ICML*, volume 97, pages 403–411, 1997.
101. D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

102. Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.
103. Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
104. Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833. ACM, 2007.
105. Jianping Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the ninth international conference on machine learning*, pages 470–479, 1992.