INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Course Project

Statistical Methods in Artificial Intelligence

Spring 2019

Intrusion Detection Techniques

Author:
Tarun Vatwani
Roll no. 2018201075
Rajat Yadav
Roll no. 2018201077
Vivek Patare
Roll no.2018201078
Ankit Mishra
Roll no. 2018201079

Supervisor:
Prof. Ravi Kiran
Sarvadevabhatla

April 29, 2019

Abstract

Intrusion Detection System (IDS) defined as a Device or software application which monitors the network or system activities and finds if there is any malicious activity occur. Outstanding growth and usage of internet raises concerns about how to communicate and protect the digital information safely. In today's world hackers use different types of attacks for getting the valuable information. Many of the intrusion detection techniques, methods and algorithms help to detect those several attacks. Objective of this project is to explore different machine learning techniques which are derived from research papers mentioned. We provide detailed implementation and analysis of different techniques and conclude which methods fare better.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A network instrusion attack can be any use of network that compromises its stability or the security of the information that is stored on computers connected to it. A very wide range of activity falls under this definition, including attempts to destabilze the network as a whole, gain unauthorized access to files or privilages, or simply mishandling and misuse of software. Added security measures can not stop all such attacks. The goal of intrusion detection is to build a system which would automatically scan network activity and detect such intrusion attacks. Core objective of the project is to explore and implement intrusion detection techniques and compare their results, computation time, etc.

We are baselining our implementations on three research papers viz.
1. ***Fast Feature Reduction in Intrusion Detection Datasets*** [1] : Paper goes into implementing classical similarity measure feature selection algorithm and proposes new method which is faster and gives comparable results.
2. ***Intrusion Detection using Neural Netowrks and Support Vector Machines***[2] : This Paper implements Neural netowork and support vector machines as approaches to intrusion detection system and compare performance of both.
3. ***Intrusion Detection with Unlabeled Data Using Clustering***[3] : This Paper considers input dataset as unlabeled dataset to to unsupervised anamoly detection. Paper presents new type of clustering technique.

For all measurements we are using KDD cup dataset which is available at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`[4]. Full dataset available has 18M records. For our training and testing purposes, we are using 10% dataset which contains 2.1M records.

Feature reduction and its relevant results are done first. other 2 papers are tested with original dataset and reduced dataset previously obtained.

Chapter 2

Data Description

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between “bad” connections, called intrusions or attacks, and “good” normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. The KDD 99 is build upon about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. The network was a simulation of military network which had three servers with rule of victim computers that exposure a lot of attacks and normal network traffic. All attacks were consisted on four main categories:

- Denial of Service (dos): Intruder tries to consume server resources as much as possible, so that normal users can’t get resources they need
- Remote to Local (r2l): Intruder has no legitimate access to victim machine but tries to gain access
- User to Root (u2r): Intruder has limited privilege access to victim machine but tries to get root privilege
- Probe: Intruder tries to gain some information about victim machine

The original raw dump network traffic were preprocessed for International Knowledge Discovery and Data Mining Tools Competition. For this, all the raw data is converted into connection. A connection means "A sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol”. This conversion is performed by Paxson algorithm. Result of this algorithm has 41 features. List of features with their description is at table 2.1

Data output has total 22 unique class values which can be subsequently mapped to 5 main categories (types of attack) as table 2.2: multirow

Feature Name	Description	Type
Duration	length (number of seconds) of the connection	continuous
protocol type	type of the protocol, e.g. tcp, udp, etc.	discrete
Service	network service on the destination, e.g., http, telnet, etc.	discrete
Src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous
hot	number of "hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of "compromised" conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
num_root	number of "root" accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a "guest"login; 0 otherwise	discrete
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have "SYN" errors	continuous
rerror_rate	% of connections that have "REJ" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have "SYN" errors	continuous
srv_rerror_rate	% of connections that have "REJ" errors	continuous
srv_diff_host_rate	% of connections that have "REJ" errors	continuous

Table 2.1: List of Features (KDD-CUP-99)

Attack	**Number of Samples**	**Category**
smurf	280790	dos
neptune	107201	dos
back	2203	dos
teardrop	979	dos
pod	264	dos
land	21	dos
normal	97277	normal
satan	1589	probe
ipsweep	1247	probe
portsweep	1040	probe
nmap	231	probe
warezclient	1020	r2l
guess_passwd	53	r2l
warezmaster	10	r2l
imap	12	r2l
ftp_write	8	r2l
multihop	7	r2l
phf	4	r2l
spy	2	r2l
buffer_overflow	30	u2r
rootkit	10	u2r
loadmodule	9	u2r
perl	3	u2r

Table 2.2: LIST OF SUBCATEGORIES IN “10% KDD

Chapter 3

Feature Reduction in Intrusion Detection Datasets

This chapter deals with different techniques for feature reduction on dataset mentioned in paper[1]. We apply results of these reduced feature dataset to predict simple naive bayes and knn classifier. Also, paper tries to invent new feature reduction technique which is significantly faster and gives comparable results. We will explain some of the most successful similarity measures such as *Correlation-Coefficients*[5], *Least square regression*[6] and *Maximal Information Compression Index*[7]

3.1 Correlation Coefficient

we use below formula to measure how much random variable x is similar to random variable y

$$Relation(x, y) = cov(x, y)/\sqrt{var(x) * var(y)} \tag{3.1}$$

where var(x) is variance of variable x and cov(x, y) is covariance of random variable x and y. This measure have following properties:

- $0 \leq 1 - |Relation(x, y)| \leq 1$
- $1 - |Relation(x, y)| = 0$ if and only if x and y are linearly related.
- Measure is symmetric
- Assume $u = (x - a)/c$ and $v = (y - b)/d$ for any arbitrary a, b, c, d we have:
  |Relation(x, y)| = |Relation(u, v)|
- Relation(x, y) is sensitive to rotation of the scatter diagram in (x, y) plane.

3.2 Least Square Regression Error

Another similarity measure that is used regularly is least square regression error or residual variance. It is the error of predicting

$$y = bx + a \tag{3.2}$$

. Where a and b are regression coefficient which can be calculated by minimizing mean square error. The error is defined as:

$$e(x, y) = var(y) * (1 - Relation(x, y)^2) \tag{3.3}$$

With this new measure if x and y have linear relation, *e(x,y)* will be 0 and if x and y have no relation at all, *e(x,y)* will be equal to 1. This measure have following properties:

- $0 \leq e(x, y) \leq$var(y)

- *e(x,y)* equal to 0 if and only if x and y have perfect linear relation
- measure is not symmetric
- error is sensitive to rotation of scatter diagram in x-y plane

3.3 Maximal Information Compression Index

This measure fixed some defects of two previous described measures. We refer to it by MICI and use minimum eigen value of of the co-variance matrix of the given two samples.

$$w'\Sigma w = w'(\lambda w) = \lambda, \tag{3.4}$$

The smallest eigen value will be 0 when features have linear relation and as the relation vanishes and the value of lambda increase. As the formula for lambda represent direction where data have the most elongation. In the other word it is the main idea behind Principle Component Analysis. PCA is based on the fact that if a data be projected along its principal component direction, it yields maximum information compaction. If we reconstruct the data, the amount of lost information is related to eigenvectors that is not considered in PCA calculation. In PCA we always use eigenvectors that have maximum corresponding eigenvalues. So the amount of lost information will be minimal.

To detect which feature should be removed from dataset using discussed similarity measures, a simple general algorithm based on K-Nearest Neighbor (KNN) algorithm is used. It has two phases. In the first phase, all of features partitioned using the similarity measure as a distance in KNN algorithm. In the second phase a feature that describes its cluster more accurately is selected as a candidate of that group.

3.4 Proposed method (Fast feature reduction)

Variance of a random variable has very high information inside. Intuitively, we used this measure to find features with low quality to be eliminated from dataset while preserving others. In the other word we used variance to discover if any feature has no class dependent information included. Features with low information should be removed. Also because of nature of intrusion detection datasets, they are very populated by samples and even a linear order for computational cost is very slow. So we tried to make our approach to be simple as possible while preserving its accuracy. We develope this fact and built upon algorithm:

- Let $X = x_i$ where i takes value 1 to N be our intrusion detection dataset, containing network connection samples containing all attack and normal connections. Each is a vector with dimesions and $x_i \in R^d$. Let $L = l_i$ with i from 1 to N be our class label for each sample. Also we have $l_i \in k$ where $k = 1, 2, ..., k$. With this definition we have k classes in our dataset and K = 5 because we use "KDD10%" as our dataset. $x_i d$ represents the value feature s = 1, 2, ..., D in connection sample i. For each class we add all data points x_i which are in same class, feature by feature.
  $S_c d = forall x_i where l_i = c | \sum x_i d / n_c$
  n_c are number of samples of class c and $S_c d$ is mean of connection sample x_i along feature d whose class is c and $c \in k$ we put all $S_c d$ in a vector and call it FS.
- Calculate variance of $S_c d$ for each feature
  $V_d = Var(S_d) E[(S_d - \mu_d)^2]$
  where μ_d is mean of all values S_d and V_d is variance of S_d
- High value in V_d represents high turbulence therefore less preferable. Sort V_d in ascending order and pick first k feature you require.

3.5 Evaluation

We use KDD99 dataset to evaluate feature reduction techniques and its effectiveness on Bayesian Network Classifier and K-Nearest Neighbor classifier and measure their accuracy.

	CC			LSRE			MICI			FFR		
feature size	10	20	30	10	20	30	10	20	30	10	20	30
Speed	0.28	0.31	0.32	1.5	1.512	1.531	1.82	1.834	1.86	0.01	0.012	0.015

Table 3.1: FEATURE SELECTION CALCULATION TIME IN SECOND.

		All	CC			LSRE			MICI			FFS		
Feature size		41	10	20	30	10	20	30	10	20	30	10	20	30
Attack type	Normal	99	32	98	95	67	98	98	90	95	98	98	98	98
	Dos	99	27	98	99	67	94	98	85	98	99	88	96	99
	Prob	90	39	67	98	67	978	97	78	88	91	30	96	97
	R2L	95	45	67	78	34	56	79	13	45	78	20	68	87
	U2R	45	20	60	70	25	34	51	0	40	52	0	40	30
overall		99	51	95	98	72	98	98	99	99	99	89	97	99

Table 3.2: EVALUATION ACCURACY RESULT OF KNN CLASSIFIER IN PRESENT .

FFR gives significantly less time for feature reduction.

Accuracy of all methods is comparably similar Therefore we can say that FFR method is optimum for data reduction for this dataset.

Chapter 4

Intrusion Detection Using Neural Networks and Support Vector Machines

So far we have done feature reduction techniques and analyzed results for KNN and Naive bayes. In this section we apply SVM and Neural network on both original and reduced dataset calculated from previous section. Paper that was referred in this section is *Intrusion Detection using Neural Networks and Support Vector Machines* [2].

4.1 SVM Approach

SVM is treated as binary classifier and is trained with all datapoints. The data is first partitioned into two classes: normal and attack, where attack represents a collection of 22 different attacks belonging to the four classes. The objective is to separate normal (1) and intrusive (0) patterns. Data is also normalized with mean 0 and variance of 1. We use standard SVM algorithm in sci-kit learn for testing with 0.2 fraction as testing data and 0.8 fraction as training data. Results are recorded with keeping parameter $C = 1$ and using linear and RBF kernal.

4.2 Neural Network Approach

Similar to SVM, Neural Network is treated as binary classification problem. Data is first partitioned into two classes: normal and attack, where attack represents a collection of 22 different attacks belonging to the four classes. Dataset is converted to all numeric values using label encoder. The objective is to separate normal (1) and intrusive (0) patterns. Data is split into 80%, 20% train-test ratio. We use following parameters to train our neural network.

- 40 nodes input layer
- 40 nodes hidden layer with sigmoid activation function
- 40 nodes hidden layer with sigmoid activation function
- output layer with 1 node (normal or attack)
- Number of epochs 100, batchsize = 32, optimized used is adam and binary cross entropy as loss function
- Class weights are also used for imbalanced data.

4.3 Our approach

Apart from above mentioned approaches, we tried to classify data as multiclass data keeping 5 output classes and training testing data used, unlike paper was full dataset for better approximation of data. Also we tested both approaches on reduced datasets obtained from feature reduction to compare results.

4.4 Results

4.4.1 SVM Results

By Keeping binary classification Dataset with full features and linear kernal. Results are:

- True Positives : 79232
- True Negatives : 19255
- False Positives : 117
- False Negatives : 200
- Accuracy : 0.9967916278693171
- Precision : 0.9985255012665566
- Recall : 0.9974821230738241
- f1 score : 0.9980035394663088

By using **RBF Kernal**, Results are:

- True Positives : 79315
- True Negatives : 19338
- False Positives : 34
- False Negatives : 117
- Accuracy : 0.99847172179264
- Precision : 0.999571513188572
- Recall : 0.9985270419981871
- f1 score : 0.9990490046038254

By Doing SVM on Binary Classification dataset with different feature reduction, results obtained are:

By Doing SVM Classification on multiclass datasets with different feature reduction, results obtained are:

4.4.2

Neural Network Results By Keeping binary classification Dataset with full features. Results on Neural Network architecture are:

- True Positives : 79215
- True Negatives : 19431
- False Positives : 42
- False Negatives : 116

- Accuracy : 0.998400874458524
- Precision : 0.9994700783527007
- Recall : 0.998537772119348
- f1 score : 0.9990037077206346

By Applying Neural Net on Binary Classification dataset with different feature reduction, results obtained are:

By Doing Neural Network on Binary Classification dataset with different feature reduction, results obtained are:

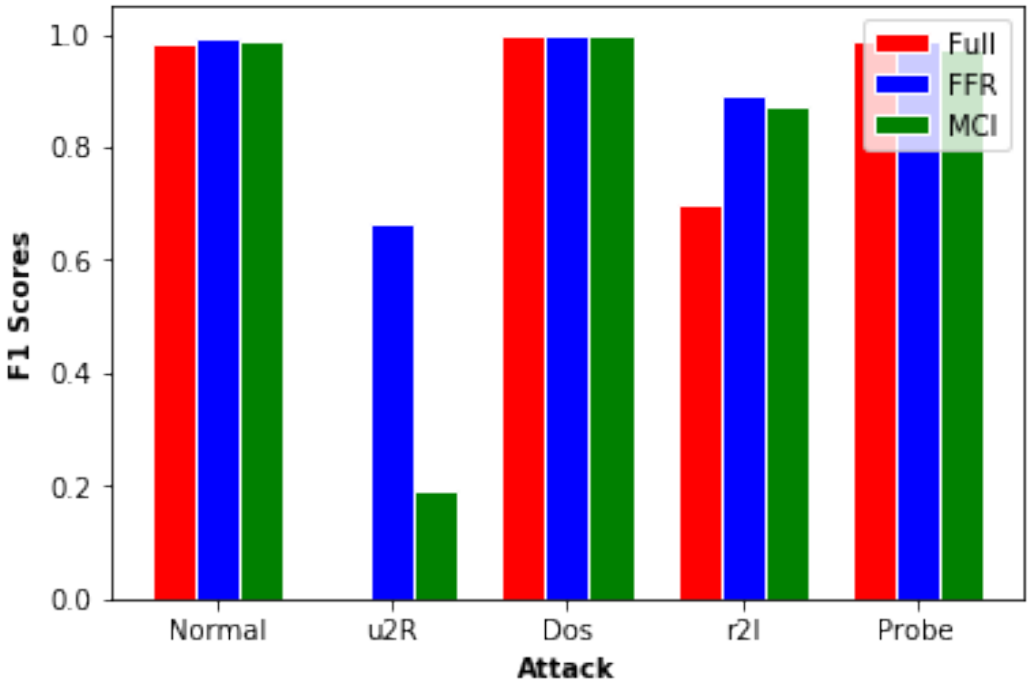

Figure 4.1: With 30 Features

Chapter 5

Intrusion Detection with Unlabeled Data Using Clustering

Anomaly detection approaches build models of normal data and then attempts to detect deviations from the normal model in observed data. Anomaly detection algorithms have the advantage that they can detect new types of intrusions, because these new intrusions, by assumption, will deviate from normal network usage . Traditional anomaly detection algorithms require a set of purely normal data from which they train their model. If the data contains some intrusions buried within the training data, the algorithm may not detect future instances of these attacks because it will assume that they are normal.

However, more often than not, we do not have either labeled or purely normal data readily available. Generally, we must deal with very large volumes of network data, and thus it is difficult and tiresome to classify it manually. We can obtain labeled data by simulating intrusions, but then we would be limited to the set of known attacks that we were able to simulate and new types of attacks occurring in the future will not be reflected in the training data. Even with manual classification, we are still limited to identifying only the known (at classification time) types of attacks, thus restricting our detection system to identifying only those types. Generating purely normal data is also very difficult in practice. If we collect raw data from a network environment, it is very hard to guarantee that there are no attacks during the time we are collecting the data

In given paper[3], we present a new type of intrusion detection algorithm, unsupervised anomaly detection (also known as anomaly detection over noisy data), to address these problems. This algorithm takes as inputs a set of unlabeled data and attempts to find intrusions buried within the data. After these intrusions are detected, we can apply train a misuse detection algorithm or a traditional anomaly detection algorithm over the data

We have to filter out data so that data follows illustrated assumptions.

5.1 Assumptions

- Anomaly samples should be very rare, such as 98.5-99% should be normal and only 1-1.5% of samples should be anomaly.
- Anomaly samples should be qualitatively different from normal samples.

After filtering out data, we have to normalise data, by subtracting mean and dividing by standard deviation, and this should be done on continuous features, categorical feature will be as it is.

5.2 clustering

To create clusters from the input data instances, we used a simple variant of single-linkage clustering Although this is not the most effective clustering algorithm, it has the advantage of working in near linear time. The algorithm starts with an empty set of clusters, and generates the clusters with a single pass through the dataset. For each new data instance retrieved from the normalized training set, it computes the distance between it and each of the centroids of the clusters in the cluster set so far. The cluster with the shortest distance is selected, and if that distance is less

than some constant W (cluster width) then the instance is assigned to that cluster. Otherwise, a new cluster is created with the instance as its center. More formally, the algorithm proceeds as follows:

Assume we have fixed a metric M, and a constant cluster width W. Let dist(C, d) where C is a cluster and d is an instance, be the distance under the metric M, between C's defining instance and d. The defining instance of a cluster is the feature vector that defines the center (in feature space) of that cluster. We refer to this defining instance as the centroid. We will be having two hyperparameters, W as width of a cluster and N as a percent of clusters classified as normal.

-
- Initialize the set of clusters S, as empty set
- Obtain a sample d from training set, if S is empty then create a cluster with d as defining instance, otherwise calculate euclidean distance from each centers(defining instances of clusters), and find a cluster C from whom d is closest.
- If distance of d from C id more than W, then make d as new defining instance of a new cluster, and insert d in S.
- Repeat 2, 3 steps until no instances are left.

We label normal and anomaly to different clusters, we will do it as, we will label N percent of largest clusters as normal, and rest as anomaly.

5.3 detection

For detection we will follow following algorithm: Given Data sample d,

- Convert d to statistical information of training set from which the clusters were created. Let d* be new sample after conversion.
- Find cluster C, which is closest to d*.
- Label d*same as the cluster C.

5.4 Performance Measure

For performance measure, instead of focusing on accuracy, we will focus on detection rate and false positive rate.

The trade-off between the false positive and detection rate is inherently present in many machine learning methods.In our system, trade-off between false positive and detection rate was very apparent. As largest cluster labelled as normal was decreased, detection rate increase substantially, since larger number of clusters are labelled as anomalous. The intrusion instances which were previously assigned to those clusters and labelled as normal, are now were classified correctly as anomalous. However, at the same time false positive rate will also increase, because all normal instances assigned to previously normal labelled clusters, are now labelled as intrusions.

5.5 Results

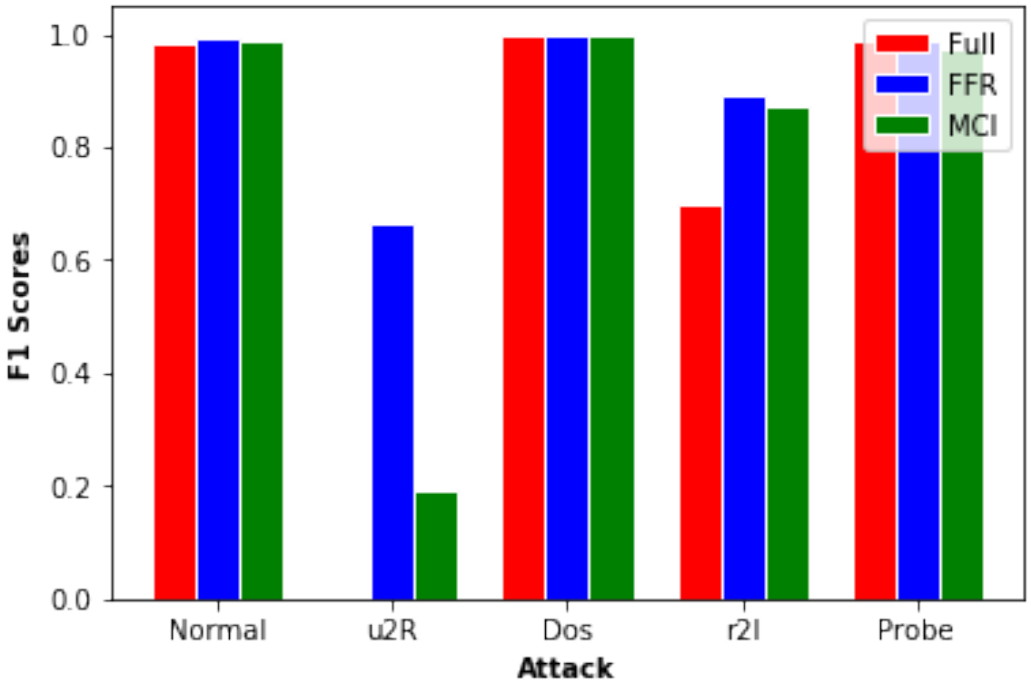

Figure 5.1: With 30 Reduced Features

Bibliography

[1] S. Parsazad, E. Saboori, and A. Allahyar. Fast feature reduction in intrusion detection datasets. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1023–1029, May 2012.

[2] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 2, pages 1702–1707 vol.2, May 2002.

[3] Leonid Portnoy, Eleazar Eskin, and Salvatore Stolfo. Intrusion detection with unlabeled data using clustering. 11 2001.

[4] Kdd cup 1999 dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[5] Thomas Lehmann, Abhijit Sovakar, Walter Schmiti, and Rudolf Repges. A comparison of similarity measures for digital subtraction radiography. *Computers in Biology and Medicine*, 27(2):151–167, 1997.

[6] J. Durbin and G. S. Watson. Testing for serial correlation in least squares regression. i. *Biometrika*, 37(3-4):409–428, 1950.

[7] Pabitra Mitra, Chaitra Murthy, and Sankar Pal. Unsupervised feature selection using feature similarity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24:301–312, 04 2002.