



UPPSALA
UNIVERSITET

UPTEC IT 14 005

Examensarbete 30 hp
April 2014

A study about fraud detection and the implementation of SUSPECT - Supervised and UnSuPervised Erlang Classifier Tool

Alexander Lindholm



UPPSALA
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet
UTH-enheten**

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
<http://www.teknat.uu.se/student>

Abstract

A study about fraud detection and the implementation of suspect - Supervised and UnSuPervised Erlang Classifier Tool

Alexander Lindholm

Fraud detection is a game of cat and mouse between companies and people trying to commit fraud. Most of the work within the area is not published due to several reasons. One of the reasons is that if a company publishes how their system works, the public will know how to evade detection. This paper describes the implementation of a proof-of-concept fraud detection system. The prototype named SUSPECT uses two different methods for fraud detection. The first one being a supervised classifier in form of an artificial neural network and the second one being an unsupervised classifier in the form of clustering with outlier detection. The two systems are compared with each other as well as with other systems within the field. The paper ends with conclusions and suggestions on how to make SUSPECT perform better.

Handledare: Bip Thelin
Ämnesgranskare: Johannes Borgström
Examinator: Lars-Åke Nordén
ISSN: 1401-5749, UPTEC IT14 005

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Problem Statement	8
1.3	Outline	8
1.4	Delimitations of scope and key assumptions	8
2	Background	9
2.1	Supervised Fraud Detection	10
2.2	Unsupervised Fraud Detection	10
3	Methods for implementation	13
3.1	Artificial Neural Networks	14
3.1.1	Artificial Neuron	14
3.1.2	Backpropagation	14
3.2	Clustering	15
3.2.1	K-means clustering	15
3.3	Outlier Analysis	16
3.3.1	Outlier Detection - the statistical approach	17
3.3.2	Outlier Detection - the distance-based approach	17
3.3.3	Outlier Detection - the density-based local outlier approach	17
3.3.4	Outlier Detection - the deviation-based approach	18
3.4	User Profiling and Scoring	18

4	Supervised and UnSupervised Erlang Classifier Tool — suspect	19
4.1	Architecture	19
4.2	suspect - supervised classifier	21
4.2.1	Training	23
4.2.2	Classification	25
4.3	suspect - unsupervised classifier	26
4.4	suspect - web GUI	27
4.5	Implementation	29
5	Performance evaluation	31
5.1	The test data	31
5.2	Testing the supervised classifier	32
5.3	Testing the unsupervised classifier	34
5.4	Results	34
5.4.1	Supervised classifier	35
5.4.2	Unsupervised classifier	35
5.4.3	A quick comparison of the results	35
6	KDDCup'99 Classifier Learning Contest	37
6.1	A comparison between the winning entry and suspect	39
7	Discussion	41
8	Future work	43
9	Summary and Conclusions	45
	Appendices	47
A	Used technologies	49
A.1	Erlang	49
A.1.1	Cowboy	49
A.1.2	Erlydtl	49
B	The KDDCup'99 test data and suspects altering of it	51

1.1 Motivation

Since the dawn of currency and private information there has been thieves and fraudsters trying to cheat and steal to gain advantage. Today with the rapid growth of the internet, the ease of access to information, the constantly new services provided and the migration of personal data and confident information to the cloud fraudsters no longer have to leave the comfort of their own home to commit fraud.

Fraud detection and fraud prevention has become a game of cat and mouse between companies and these fraudsters. The techniques are constantly changing and companies need to react and stay one step ahead. Today all major companies that serve delicate information need to have a fraud detection system. Normal fraud prevention techniques like password protection are not enough. Different techniques used for fraud detection have been proposed throughout the years and are normally within the fields of datamining and statistics, but artificial intelligence and machine learning are fields that also are useful in the area.

Kivra is a company that offers other companies the possibility to digitally send information that would have normally been sent by postal services. Their service is a cheaper and more environmental-friendly way for companies to communicate with their customers. Kivra's service allows end users to receive, handle and archive invoices, documents and other typical postal items at home in their browser. A very important part of the work of Kivra's platform is security.

1.2 Problem Statement

The purpose of this thesis is to in collaboration with Kivra develop a fraud detection framework that might come to be used with Kivra's current system. The framework's task is to determine if a user still can be trusted or if it should be flagged for potential fraud. This thesis starts with an academic study that looks at previous studies and research about fraud detection. The purpose of the academic study is to find existing algorithms, methods but also to see what kind of target variables that are being examined in existing implementations and systems.

With the knowledge gained from the academic study a prototype is implemented. After the implementation the prototype is tested to see how well the algorithms and implementation performs.

1.3 Outline

The thesis starts with an introductory chapter (Chapter 1) and then continues with the second chapter which is intended to give the reader the necessary background knowledge needed for understanding the concepts and ideas around the implementation mentioned in the later chapters of this thesis. The third chapter describes the methods that will be used and implemented in the prototype described in chapter four. Chapter five, six and seven concerns testing, results and comparisons of SUSPECT with other systems. In the seventh chapter we discuss the subjects from the previous chapters. In chapter eight future work is discussed. The last chapter, chapter nine, contains a summary and conclusions regarding the thesis.

1.4 Delimitations of scope and key assumptions

The prototype developed in this thesis is supposed to be able to work with Kivra's platform, which has some impact on the decisions regarding what tools to use and which methods and/or algorithms that are chosen. Due to the time limit of this thesis, the fraud detection system developed will not be a finished product ready to ship, but rather a proof-of-concept prototype. Data from the system that this prototype is designed to be in use with was not provided at the start of this thesis so external data will be used, which has some impact on which algorithms and methods that will be used.

Background

One has to distinguish Fraud Prevention and Fraud Detection (Bolton, 2002).

Fraud prevention works proactively, striving to stop fraud before it occurs. Everyday examples of this would be passwords, watermarks, holographs on banknotes [4]. These examples are actions taken to prevent fraud before they occur, but fraud prevention is not flawless peoples passwords can get stolen and credit cards can get skimmed. This is where fraud detection comes in.

Fraud detection works reactively and is responsible for detecting fraud once fraud prevention has failed. Fraud detection systems need to be continuously developed due to the fact that if a fraudster knows a fraud detection system is in place he/she will figure out new ways to work around it. One also has to keep in mind that new fraudsters will come along and there is a chance that they will use techniques tried by other fraudsters earlier and therefore one can not remove earlier detection methods but rather have to extend the old one [4]. Under the hood fraud detection is classification of data, and as with classification this can be done with several approaches of the types supervised, reinforced or unsupervised.

One problem with the development of a new fraud detection system is that the number of ideas published is limited. This is due to that researchers and companies rather neither publish their fraud detection techniques in detail, nor the data sets used in detail, since this invites the general public to know how to evade detection. The work within the field of fraud detection that has been published is mostly focused on supervised systems: systems that needs to be provided with examples of fraudulent and non-fraudulent behaviour to

be able to classify new entries.

Statistical tools used for fraud detection are many and varied due to different data types and sizes, but the majority is working by comparing the observed data with expected values [4]. Statistical fraud detection methods can generally be divided into supervised and unsupervised methods, both described briefly in 2.1 and 2.2. Methods used are everything from traditional statistical classification methods [9] [18] such as discriminant analysis but also more powerful tools such as artificial neural networks (Hand 1997; Webb, 1999) has proven successful. Rule-based methods have been applied extensively within supervised fraud detection, examples are rule-trees [5], naive Bayesian [25] as well as C4.5 [22]. These algorithms can be described as a series of "if then else" cases.

In contrast to supervised fraud detection, unsupervised fraud detection does not need examples of fraudulent and non-fraudulent behaviour but rather works by finding entries that is looking somewhat dissimilar to the rest of the dataset.

2.1 Supervised Fraud Detection

A supervised method requires samples of both good and bad records to assign new objects into one of these two categories. Linear discriminant analysis [9] and logic discrimination [9] are examples of tools that have proven themselves to be effective, but more powerful tools such as artificial neural networks are often applied as well [4]. More about artificial neural networks can be read in Section 3.1.

However, since these methods are taught what is considered fraudulent and non fraudulent, they are limited and will in most cases only detect the types of fraud that has been committed before (since what one teaches the method to be fraudulent is often previously occurred frauds).

2.2 Unsupervised Fraud Detection

In contrast to supervised statistical methods, unsupervised statistical methods does not use samples of good and bad records to distinguish fraud but rather uses larger sets of data for profiling and outlier detection: detection of objects that don't comply with the rest of the dataset. More about outliers can be found in Section 3.3. That an object, or user, is flagged as an

outlier doesn't mean that it is an attempt of fraud, it just means that the object doesn't look like the rest of the set and should be further examined. This leads us to note that statistical analyses alone can seldom decide if a fraud has been perpetrated [3]. What the statistical analysis returns can be regarded as a suspicion score [3] (where a higher score is regarded to be more suspicious than a lower one). The higher the score is, the less the object conforms to the rest of the data set and the more it should be further investigated. An advantage with unsupervised fraud detection over supervised is that it might discover previously undiscovered types of fraud [16], since supervised fraud detection works based on known sets of fraudulent and non fraudulent records.

Methods for implementation

In Chapter one it was stated that one or two methods for fraud detection would be chosen for implementation. In Chapter two the difference between supervised fraud detection and unsupervised fraud detection was described. In this chapter we describe the two methods that are chosen, one unsupervised and one supervised. As mentioned, supervised methods are more widely used in the field of fraud detection, but unsupervised methods have some potential. So one from each is chosen for implementation so the performance can be compared.

Methods that are very domain specific such as decision-trees will not be chosen since the data we will use for the testing of SUSPECT is not from the same system as SUSPECT is supposed to work with. Methods that are easily customizable are chosen. For the supervised classifier an artificial neural network will be created and for the unsupervised classifier clustering with outlier detection will be created.

Artificial Neural Networks (ANN) is chosen as our supervised algorithm due to its ability to derive meaning from complicated data and detect trends that are too complex to be noticed by humans or more naive techniques. ANNs have also been proposed for use in fraud detection systems in several articles [8, 7, 15] .

Outlier detection is a reappearing subject within the field of unsupervised fraud detection [3, 4]. The unsupervised algorithm used will be k-means clustering with outlier detection. K-means clustering will be used since the data we will be testing with is not grouped and k-means is good at grouping data and easy to implement [14].

3.1 Artificial Neural Networks

The roots of neural networks can be traced back to neuro biological studies from over a century ago. Back then the research was mainly about how the nervous system worked. 1890, William James made a statement that is reflected in the subsequent work of many researchers [19].

The amount of activity at any given point in the brain cortex is the sum of the tendencies of all other points to discharge into it, such tendencies being proportionate

- to the number of times the excitement of other points may have accompanied that of the point in question;
- to the intensities of such excitements; and
- to the absence of any rival point functionally disconnected with the first point, into which the discharges may be diverted.

A little more than fifty years later, in 1943, McCulloch and Pitts presents the McCulloch-Pitts (MCP) neuron, which is the first mathematical model of a single neuron [19]. Over time the MCP neuron has been modified and widely applied in subsequent work.

3.1.1 Artificial Neuron

Simply put an ANN is built up of neurons which are connected to each other with synapses (incoming connections) and dendrites (outgoing connections). Each of these connections contains weights, and upon stimulation the neuron forwards an output. What the output looks like depends on the input and the weights of the synapses. These neurons are then trained for the task at hand by a either supervised-, unsupervised- or reinforced-learning method. The training itself is adjustment of weights for all synapses of the neuron. Supervised learning methods can be divided into two general groups, Error Correction Rules and Gradient-descent-based rules. More about this can be found in [19] and [10].

3.1.2 Backpropagation

Backpropagation is without a doubt the most widely used architecture for artificial neural networks. The reason for this is backpropagations ability to learn complex multidimensional mappings [11]. How the ANN is trained with backpropagation can be read in Section 4.2.1.

3.2 Clustering

Clustering, also known as cluster analysis, is the art of grouping a set of data objects into several subgroups, called clusters, with each cluster containing data entries that are similar to the other data entries in that cluster. Clustering sees extensive use in the data mining area and is also a common technique for statistical data analysis in fields such as machine learning, pattern recognition and image analysis. Different researchers use different clustering algorithms depending on the application, such as

- Connectivity models, hierarchical clustering models.
- Centroid models, where each cluster is represented by a single mean vector.
- Density models, clusters are defined by how dense regions in the data space are.

These clustering models and many more can be read about in [6].

3.2.1 K-means clustering

K-means clustering is a centroid clustering model that works by dividing the n items in a dataset into k clusters. K-means is computationally difficult (NP-Hard) but there is efficient heuristic algorithms which converge quickly to a local optimum. Bacher, 2004, suggests two algorithms for k-means, one being a serial and the other being a parallelized (see Algorithm 1).

The parallelized algorithm is used for clustering in the developed prototype.

Algorithm 1. Parallel k-means (PKM)

Input: set of data, n , and number of clusters, k
Output: k -centroids and members of each cluster
 Select initial centroid $C = \langle C_1, C_2, \dots, C_K \rangle$
 Partition data to P subgroups, each subgroup has equal size
foreach P **do**
 Create a new process
 Send C to the created process for calculating
 distances and assigning cluster members
 Receive cluster members of K clusters from P processes
 Recalculate new centroid C'
 if $\text{difference}(C, C')$ **then**
 | Set C to be C' and go back to step 2;
 end
 else
 | Stop and return C as well as cluster members
 end
end

3.3 Outlier Analysis

Outlier Analysis, or Outlier Mining, is a way to extract data that differs from the trend of the remaining data. These outliers can be interpreted as noise in some systems and are often discarded. Although in the field of fraud detection these objects might be the most interesting ones to look at. Objects that don't comply to the rest of the data. Users that don't behave like the rest of the users.

In [13] outlier mining is described as follows: in a set of n data points and given k , the expected amount of outliers, find the top objects that are dissimilar or inconsistent with respect to the rest of the data. According to [13], this process can be divided into two subproblems:

- "Defining what data to be considered as inconsistent in the given data set"
- "Find an efficient method to mine the outliers so defined."

The human eye can easily see inconsistency within data visualization but only to some extent. The problem with this approach is when cyclic patterns occur, this results in that the eye can misinterpret an outlier as a normal user and vice versa [13].

A more reliable way of finding outliers is with the help of mathematical methods of applied statistics. There are four commonly used computer-based methods for outlier detection which will be briefly explained in the next subsections.

3.3.1 Outlier Detection - the statistical approach

This approach is commonly used to find outliers when there is only one variable of interest [20]. The approach takes advantage of a probabilistic data model along with mathematical methods of applied statistics. By constructing this probabilistic data model outliers can be found by a discordancy test [13].

The downside with this approach is that one needs knowledge about the distribution parameters and in some extent need to know the expected amount of outliers. Statistical methods does not guarantee that all outliers will be found for the cases where no special test was developed [13].

3.3.2 Outlier Detection - the distance-based approach

The distance-based approach is to prefer over the statistical approach if the dataset you want to investigate has an unknown amount of outliers [20]. This approach looks at all the objects within the dataset and sorts out the objects with the least amount of "neighbours".

An outlier in the distance-based approach, $DB(pct, dMin)$ is defined in the following way: An object o is an outlier if at least a pct fraction of all objects in the dataset are at greater distance than $dMin$ from the object o . Several algorithms for mining distance-based outliers have been developed such as the Index-based algorithm [23] and the Nestled-loop algorithm [23], see [13, 20, 23]. Distance-based outlier detection involves that the user has to set both pct and $dMin$, which involves a lot of trial and error before getting good results.

3.3.3 Outlier Detection - the density-based local outlier approach

Both previously mentioned approaches, the statistical- and the distance-based is operating on an overall or "global" distribution of data points [13]. Keep in mind that user data might change with trends, and the amount of login attempts might increase drastically during peak hours. In distance-based approach it was stated that it takes some trial and error before getting a good pct and $dMin$ values. That is because of the spreading of the objects.

That problem is solved in this approach thanks to the introduction of "local outliers". The key to this approach is that deciding if an object is an outlier or not is no longer binary. Every object rather have an outlier degree. This degree is calculated as the objects Local Outlier Factor (LOF) [13].

3.3.4 Outlier Detection - the deviation-based approach

The deviation-based approach does neither use statistical methods nor distance-based measures to find outliers. Rather it examines and compares the main characteristics of the group to find objects that deviate from the rest. Two common techniques used of deviation-based outlier detection is the Sequential Exception Technique and OLAP Data Cube Technique, more about these can be found in [13].

3.4 User Profiling and Scoring

In computer security a user profile is a collection of characteristics of a user such as login location, typing rate, counts of particular commands [12]. Depending on the purpose of the system different kinds of user characteristics can be stored within a user profile [24]. These profiles can later be used for scoring purposes, such as risk scoring or identity scoring. A few examples of characteristics that can be used for risk scoring are [21]:

- Invalid logins before a transaction.
- Idle account has suddenly become active.
- Transactions have happened in the past hours during which the customer has applied for card replacement.
- Transactions with unusual time stamps or from black- listed geographical locations.
- Inter-country or inter-currency transactions of high amounts.
- New or unknown payee as per historical records.

These kinds of variables, or user actions, can be compared to predefined boundaries or other user profiles to flag a user to be fraudulent or non fraudulent. How one compares the variables depends on the implementation and method in use.

Supervised and UnSupervised Erlang Classifier Tool — suspect

During this thesis a prototype for fraud detection was implemented. The system consists of three main parts, a unsupervised classifier, a supervised classifier and a web-GUI.

4.1 Architecture

SUSPECT is mainly written in Erlang and as mentioned it consists of four main parts. A supervised classifier consisting of an ANN. An unsupervised classifier consisting of k-means clustering with outlier detection. A web-GUI and a database where previous classifications are stored. The web-GUI is hosted with the Cowboy HTTP server ¹ and serves DTL (Django Template Language)² pages with the help of ErlyDTL³. The database in the prototype is Mnesia⁴. Mnesia was chosen as the database storage due to its ease to store and retrieve data in Erlang. Communication to the supervised classifier and to the unsupervised classifier is done by Erlang messages. Communication with the supervised- and unsupervised classifier is done with Erlang messages due to the fact that the system SUSPECT is supposed to work with is mainly coded in Erlang.

¹<https://github.com/extend/cowboy>

²<https://docs.djangoproject.com/en/dev/ref/templates/builtins/>

³<https://github.com/erlydtl/erlydtl>

⁴http://www.erlang.org/doc/apps/mnesia/Mnesia_chap1.html

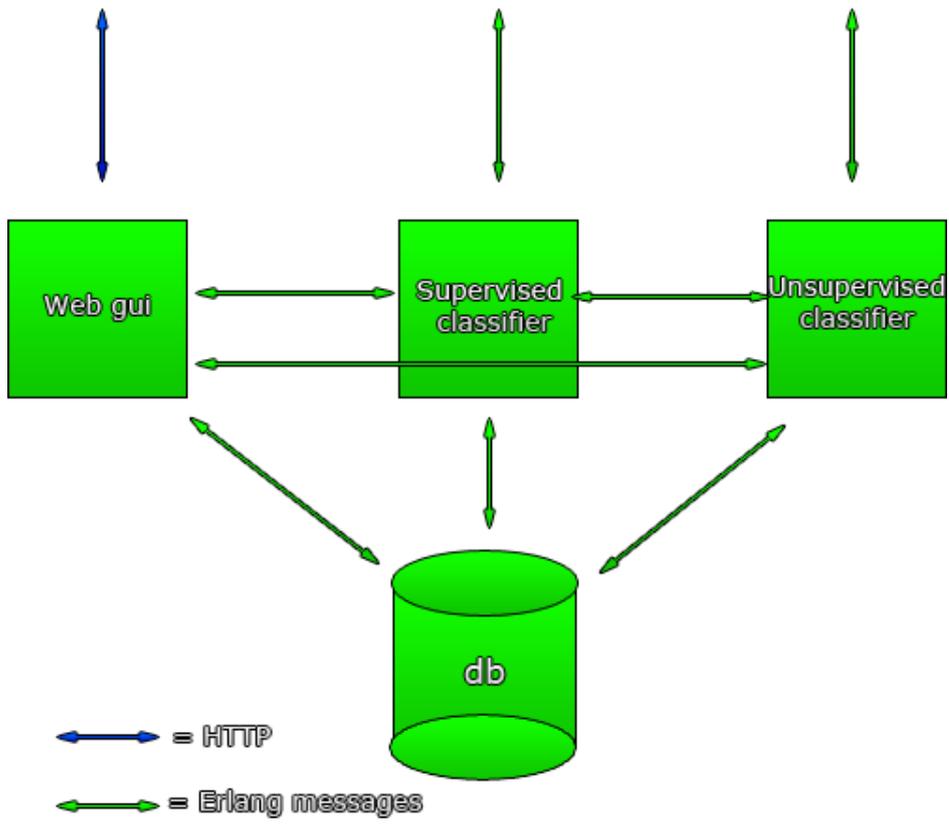


Figure 4.1: The architecture of suspect and how they communicate

4.2 suspect - supervised classifier

The supervised classifier consists of an ANN along with a trainer. The ANN is built up of three layers of neurons, an input layer, a hidden layer and an output layer. The neurons are spawned as individual Erlang processes which keeps track of which neurons it is connected to. Each node on each layer is connected to each node on the next. For every incoming connection each neuron contains a weight which determines how it interprets the input received from the connected neuron.

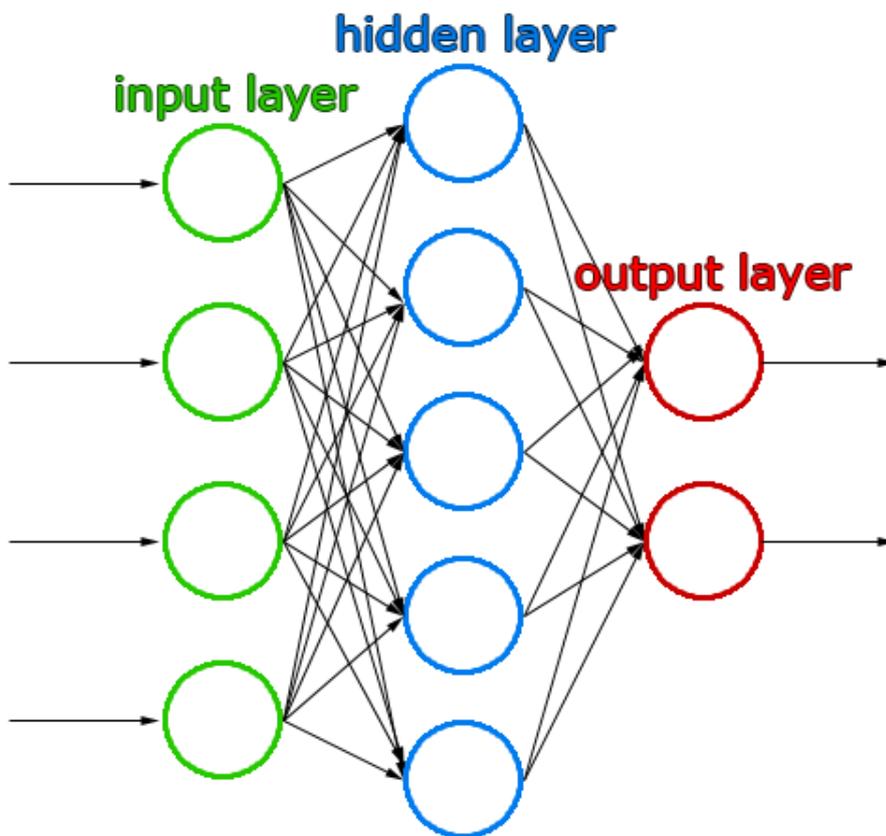


Figure 4.2: A small ANN consisting of 4 input neurons, 5 hidden neurons and 2 output neurons along with their respective connections.

Each of these neurons does two actions upon stimulation before sending its prediction to its outputs, the first being summing the inputs along with the corresponding weight. The second being sending the sum through a nonlinear function called the neuron activation function.

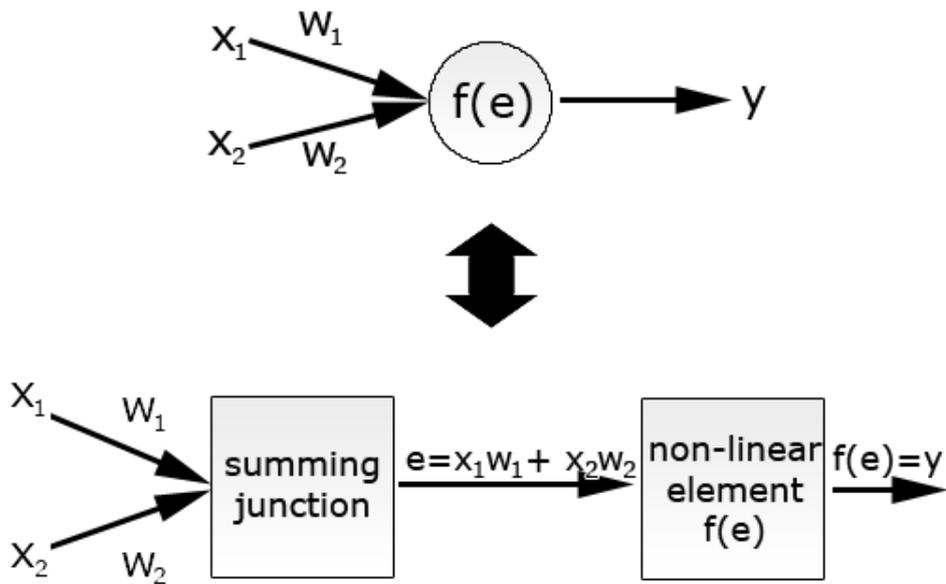


Figure 4.3: Anatomy of a single neuron with two inputs

4.2.1 Training

Before using the SUSPECT supervised classifier, it needs to be taught what is considered to be fraudulent and what is not. Training works by giving the trainer a path to a file containing test sets in form of an CSV (Comma separated values) file. Answers to each line need to be included as the last element of each line. The trainer starts by reading the first line and separates the answer from the rest of the values and then feeds these values to the input nodes of the ANN. Each input node gets one value. The input nodes then forward everything to the hidden layer. The hidden layer then alters the values it gets from its input nodes according to their respective weights (see Figure 4.3). When that is done all the nodes in the hidden layer will send the new output to the output layer which performs the same action as the hidden layer. Once all values have reached the output nodes and their calculations are done, the ANN outputs a prediction.

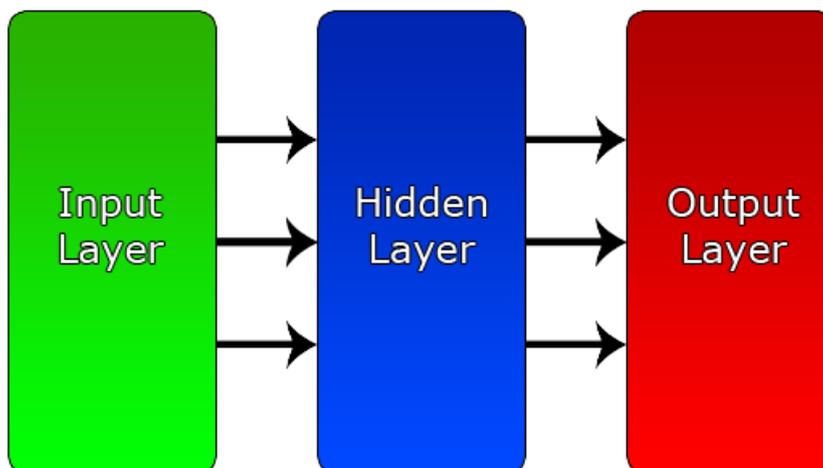


Figure 4.4: Each neuron in the input layer forwards its prediction to each neuron in the hidden layer. Each node in the hidden layer then forwards its prediction to each neuron in the output layer.

Once the output layer has done a prediction, the trainer looks up the answer for the entry and tells each output neuron how far off it was, each output neuron then does changes to the weights accordingly and starts to backpropagate the error so that the nodes in the hidden layer can adjust its weights accordingly as well.

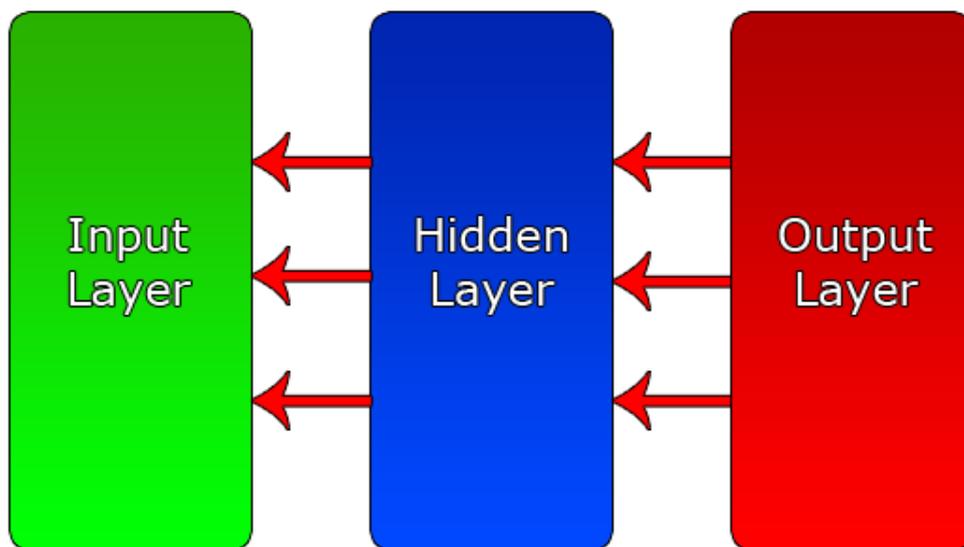


Figure 4.5: Each neuron in the output layer backpropagates the error to each neuron in the hidden layer.

When this is done, the trainer takes the next training set and sends it into the network the same way as the one before. This is repeated until the trainer reads end of file from the sent in CSV file. After training is done the ANN is ready for classification of new entries.

4.2.2 Classification

When the ANN is trained, it can be used for classification and prediction. This works by sending the ANN a list containing all the values to be evaluated. The ANN then does the same procedure as the first part of the training, forwarding everything through the network and all nodes adjust the values accordingly. But after forwarding, instead of doing backpropagation, the ANN returns the score that the entry got. This is compared to the current threshold, which is computed when the ANN is trained, to determine if the entry evaluated should be considered fraudulent or non-fraudulent.

4.3 suspect - unsupervised classifier

The unsupervised classifier works by clustering the received data with the k-means clustering algorithm and then looks how far off the entry is from the closest center to determine if the entry can be potential fraud or not. The number of centers needs to be determined before starting using the unsupervised classifier. This is due to the nature of the k-means algorithm. The k-means algorithm used is a parallelized version, which can be read more about in 3.2.1.

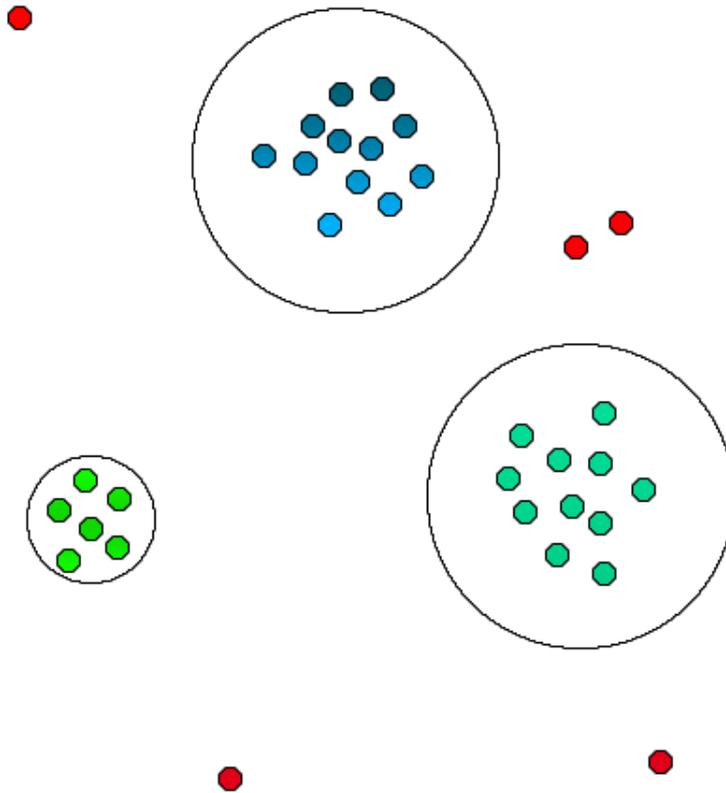


Figure 4.6: The circled dots are entries belonging to clusters and the ones outside are potential fraud

4.4 suspect - web GUI

SUSPECT contains a web GUI that can be used to see old entries, previous classifications, detailed data about the previous entries and status of the classifiers. It serves HTML pages with the help of an Erlang back end. The open-sourced http server "cowboy" is used for all communication done via HTTP and the open-source ErlyDTL ⁵ is used for compilation of DTL to Erlang bytecode. For swift development and cross-browser compatibility Twitter bootstrap⁶ was used.

The GUI mainly shows a table with four columns: id, prediction, score and answer (see figure 4.7 and figure 4.8). The prediction column is the prediction that SUSPECT has made for the entry. The score is the score that the entry got. The answer column can have the values 0, 1 or unknown where values 0 and 1 represents if the system were told what the answer for the entry was and unknown is used for entries without answers. The rows are color coded. A green row means that SUSPECT has classified that entry to be non-fraudulent and a red row means that SUSPECT has classified that entry to be fraudulent.

The score seen in the third column is what determines if SUSPECT classifies the current entry as fraudulent or not, its the value that gets compared to the threshold. This threshold is set and adjusted by SUSPECT during training.

⁵<https://github.com/evanmiller/erlydtl>

⁶<http://getbootstrap.com/>

id	prediction	score	answer
617985927	suspect	0.9734318605711936	1
184226394	suspect	0.9707041740682077	1
1219552147	suspect	0.8746322118073865	1
1005107396	suspect	0.9837473302805428	1
984523380	suspect	0.8864399984925029	1
373538380	suspect	0.9783472597255322	1
1172091656	suspect	0.9944110195547549	1
23805074	suspect	0.9414281586778233	1
399442771	suspect	0.9770072332341856	1
560928166	suspect	0.9948375492291541	1
277893827	suspect	0.9771908897623729	1
85487099	suspect	0.9567481268035299	1
1075729024	suspect	0.8708705953253095	1
496895565	suspect	0.9772460619366	1
774501062	suspect	0.9783592555992391	1
837666673	suspect	0.979908079990277	1

Figure 4.7: Suspect web GUI screenshot with correctly classified fraud entries

id	prediction	score	answer
67344277	ok	0.7469043477397782	0
626516376	ok	0.25218613493766023	0
588629814	ok	0.24704235596831342	0
285419240	suspect	0.8998492942871781	0
563475081	ok	0.24516423345559607	0
468055700	ok	0.23528848256710838	0
130564062	ok	0.8212270647479193	0
1351402710	ok	0.7064199994138632	0
218666726	suspect	0.8800777185071397	0
37504868	ok	0.7624350710748914	0
403748499	ok	0.23010695140959908	0
438251989	ok	0.28232292543684206	0
12487435	ok	0.7416375408888652	0
1323989392	ok	0.7272972984582018	0
156966852	ok	0.8364066106174025	0
317226158	suspect	0.9216528265554894	0
254250767	suspect	0.890737427774991	0

Figure 4.8: Suspect web GUI screenshot with correctly classified and misclassified non-fraud entries

4.5 Implementation

The system contains of several Erlang modules along with a few DTL templates. The three main parts, the supervised classifier, the unsupervised classifier and the web-GUI are made up of three individual `gen_servers`⁷ along with their respective supervisor. Each part of SUSPECT is separate and can be placed on different servers if so desired.

⁷http://www.erlang.org/doc/man/gen_server.html

Performance evaluation

The data used for training the classifiers is domain specific and it is essential that it is well formed for the system to be able to classify new entries. Since no data was provided from the system SUSPECT is supposed to work with, external data was used for testing and evaluation.

5.1 The test data

The systems was tested with test data from the KDD Cup 1999 data and Intrusion Detection data from DARPA [1]. The data consists of several CSV files. Every row consists of 42 different elements, one of them being the classification of the entry.

The different classification of the data are: back, buffer_overflow, ftp_write, guess_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod, portsweep, rootkit, satan, smurf, spy, teardrop, warez-client and warezmaster.

All these classifications are belonging to one of five different categories: normal, probe, dos, u2r or r2l. Which classification that belong to what category can be seen in Appendix B. During the testing of SUSPECT, all classifications except "normal" were considered to be fraudulent due to the purpose of this prototype being classifying an object as either fraudulent or non-fraudulent.

Hidden nodes	tp	fn	tn	fp	$fp - rate$	$fn - rate$	Acc.
20	531	815	4103	551	17.5%	60.5%	77.2%
30	757	589	4172	482	12.6%	43.7%	82.1%
40	814	532	4296	358	11.4%	39.5%	85.1%
50	1073	273	4325	329	5.8%	20.9%	89.9%
60	1051	295	4258	396	6.3%	21.9%	88.4%

Table 5.1: Accuracy of the ANN with different numbers of hidden nodes. All setups have 41 input nodes and 2 output nodes

5.2 Testing the supervised classifier

Test #1 – Number of neurons

The number of nodes of each layer needs to be specified for the supervised classifier to be able to train and classify new entries. Two randomly chosen smaller subsets of 6000 entries from the data described in Section 5.1 is used for training and testing of different sizes of ANN’s to find the most suitable ANN setup for the data at hand. The first subset is used for training and the second for testing. The distribution of the test set is 4654 entries which should be considered non-fraudulent and 1346 entries which should be considered fraudulent.

To measure the performance of different setups of the ANN we use fp false positives, fn false negatives, tp true positives, tn true negatives to calculate the Accuracy, $Acc. = \frac{tp+tn}{tp+fp+fn+tn}$, of each setup. For each setup we use 41 input nodes and two output nodes, since the data has 41 inputs and the output can either be fraudulent or non-fraudulent. We use different amounts of hidden nodes to see which setup that gives us the best accuracy. As seen in Table 5.1, the setup 41-50-2 gives us the highest accuracy for the data at hand with an accuracy of 89.9%, $fn - rate$ of 20.9% and $fp - rate$ of 5.8%. With fewer hidden nodes the supervised classification outputs more false negatives and false positives. The system gets better accuracy as the number of hidden nodes increases until we reach 60 hidden nodes where the accuracy starts to diminish again.

Test #2 – Size of Training- and Test-data

The ANN setup of 41 input nodes, 50 hidden nodes and 2 output nodes, is fed with different sizes of training- and test-sets to see how the size of the training set affects the classification of the supervised classifier. The

Training-set Size	Training time
200	0min 8sec
400	0min 22sec
1000	5min 14sec
2000	8min 39sec
6000	55min 52sec

Table 5.2: Training times on a PC with an i5-3570k CPU and 4gb ram.

Training-set Size	tp	fn	tn	fp	$fp - rate$	$fn - rate$	Acc.
200	563	783	3418	1236	26.5%	58.1%	66.3%
400	582	764	3326	1328	28.5%	56.7%	65.1%
1000	710	636	3893	761	16.4%	47.2%	76.7%
2000	832	514	4119	535	11.4%	38.1%	82.5%
6000	1073	273	4325	329	5.8%	20.9%	89.9%

Table 5.3: Accuracy of the ANN with different sizes of training sets.

training- and test-sets are randomly chosen chunks of different sizes from the data from Section 5.1. The same test-set as used in Test #1 is used, the one that contains 6000 entries (4653 negative and 1346 positive entries).

k	tp	fn	tn	fp	$fp - rate$	$fn - rate$	Acc.
40	564	782	2934	1720	36.9%	58.1%	58.3%
60	782	564	3067	1587	34.0%	41.9%	64.1%
80	763	583	2978	1676	36.1%	43.3%	62.2%
100	434	912	2735	1919	41.1%	67.7%	52.3%

Table 5.4: Accuracy of the unsupervised classifier with different amount of clusters, k .

5.3 Testing the unsupervised classifier

Test #3 – Number of clusters, k

The number of clusters, k , needs to be specified for the unsupervised classifier to be able to classify new entries. The same test-sets used to test the supervised classifier in section 5.2 is used to test the unsupervised classifier. The distribution of the test set is 4654 entries which should be considered non-fraudulent and 1346 entries which should be considered fraudulent.

To know which value k to begin with, we take use the clustering Rule of Thumb[17].

To measure the performance with different amount of clusters we use the same measurements as in section 5.2; fp false positives, fn false negatives, tp true positives, tn true negatives and accuracy.

As seen in Table 5.4, the setup of $k = 60$ gives us the highest accuracy for the data at hand with an accuracy of 64.1%, $fn - rate$ of 41.9% and $fp - rate$ of 34.0%. With fewer clusters the unsupervised classifier outputs more false negatives and false positives. After 60 clusters the accuracy starts to diminish again.

5.4 Results

The SUSPECT prototype was tested and evaluated with sets from the test data described in 5.1. The goal of the evaluation is to see the prototype's ability to classify fraud and see how much of the data that is misclassified (false-positives and false-negatives). Since SUSPECT contains two subsystems they both were tested separately.

	<i>Supervised</i>	<i>Unsupervised</i>
<i>Acc.</i>	89.9%	64.1%
<i>fn – rate</i>	20.9%	41.9%
<i>fp – rate</i>	5.8%	34.0%

Table 5.5: Accuracy, false-negative rate and false-positive rate from the best run of the supervised- and the unsupervised classifier.

5.4.1 Supervised classifier

As seen in Section 5.2 and the Tables 5.1 and 5.3 the results of SUSPECT’s supervised classifier look promising. The results differed a bit depending on the size of the training set and the quiz set but also on how large portions of those sets that were fraudulent and non-fraudulent. The supervised classifier reached an accuracy of 89.9%, *fn – rate* of 20.9% and *fp – rate* of 5.8% with the setup of 41 input nodes, 50 hidden nodes and two output nodes. The training set for that run was 6000 entries.

5.4.2 Unsupervised classifier

As seen in Section 5.3 and the Table 5.4, the unsupervised classifier has worse performance than the supervised. The results differs a bit depending on the number of clusters. The best run in table 5.4 had an *fn-rate* of 41.9%, *fp-rate* of 34% and an accuracy of 64.1%.

5.4.3 A quick comparison of the results

Both system were evaluated with the same data set. The best run with the supervised classifier resulted in an accuracy of 89.9%, *fn-rate* of 20.9% and *fp-rate* of 5.8%. The best run with the unsupervised classifier resulted in an accuracy of 64.1%, *fn-rate* of 41.9% and *fp-rate* of 34%. When comparing the two, we see that the supervised had an accuracy rate of around 25% higher than the unsupervised. The supervised also had an *fn-rate* that was around 20% lower and the *fp-rate* was around 30% lower (see Table 5.5).

KDDCup'99 Classifier Learning Contest

The data used for testing of the SUSPECT prototype was taken mainly from the KDDCup'99 Classifier Learning Contest [1]. There were 24 teams that competed in the contest. There are several different ways to measure the performance of intrusion detection systems, among those are; detection rate (accuracy), false alarm rate and cost of misclassification. In the KDDCup'99 Classifier Learning Contest, the submitted testers were judged by the cost of misclassification.

The total misclassification cost (TMC) or average misclassification cost (AMC) can be computed with the misclassification cost matrix (see table 6.1) and the confusion matrix for a testers entry.

$$TMC = \sum_{x,y} CM(x,y) \times C(y),$$
$$AMC = \frac{TMC}{\sum_{x,y} CM(x,y)}$$

The winning entry achieved an average misclassification cost of 0.2331 per test example with the confusion matrix seen in table 6.2.

	<i>Normal</i>	<i>PROBE</i>	<i>DOS</i>	<i>U2R</i>	<i>R2L</i>
Normal	0	1	2	2	2
PROBE	1	0	2	2	2
DOS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Table 6.1: Misclassification cost matrix for the KDDCup’99 Classifier Learning Contest [1].

	<i>Normal</i>	<i>PROBE</i>	<i>DOS</i>	<i>U2R</i>	<i>R2L</i>	<i>Accuracy</i>
Normal	60,262	243	78	4	6	99.45%
PROBE	511	3,471	184	0	0	83.32%
DOS	5,299	1,328	233,226	0	0	97.12%
U2R	168	20	0	30	10	13.16%
R2L	14,527	294	0	8	1,360	8.4%
FP-rate	25.39%	35.19%	0.12%	28.57%	1.16%	
Total misclassification cost = 72,500, false-negative rate = 8.19%, Cost per example = 0.2331						

Table 6.2: Winning entry of the KDD’99 Classifier Learning Contest [2].

	<i>Sup.</i>	<i>Unsup.</i>	<i>Winner</i>
<i>Acc.</i>	89.9%	64.1%	96.5%
<i>fn - rate</i>	20.9%	41.9%	8.19%
<i>fp - rate</i>	5.8%	34.0%	26.3%

Table 6.3: Overall accuracy, false-negative rate and false-positive rate from the best run of the supervised and the unsupervised classifier of SUSPECT as well as from the winner of the KDDCup’99

6.1 A comparison between the winning entry and suspect

A fair comparison between the two is not applicable due to several reasons; SUSPECT has not yet done a successful run of the whole test set, therefore it has not been as trained as the winning entry. SUSPECT was also created to make the assumption that all negative-examples are considered as the same type, this results in that the misclassification cost matrix from Table 6.1 is not applicable on SUSPECT's results since this requires the data to be classified into five different groups, while SUSPECT only classifies entries as fraudulent or non-fraudulent.

Table 6.3 contains the accuracy, fp-rate and fn-rate from SUSPECT as well as an assembled accuracy, assembled fp-rate and fn-rate from the winner of the KDDCup'99. The winner of the KDDCup'99 has better accuracy and a better fn-rate than both parts of SUSPECT. The supervised part of SUSPECT has a better fp-rate than both the unsupervised part of SUSPECT as well as the winner of the KDDCup'99. Although these numbers are not totally reliable due to the circumstances described earlier.

SUSPECT does only use a small fraction of the data that the winner of the kddcup'99 uses and that is the reason that the numbers from Table 6.3 are not reliable. The fact that SUSPECT only categorizes entries into two groups while the others categorize into five has an impact on the accuracy, fp-rate as well as fn-rate.

Another drawback of SUSPECT is its incapability to handle discrete values. The data from the KDDCup'99 contains several fields such as IP- number, port number, protocol type etc, which SUSPECT just converts into decimal values. These decimal values are then used for computation just as continuous variables which could result in training which harms the system since the values are just made up and supposed to be discrete.

Discussion

This thesis starts off with an academic study of existing fraud detection systems and what kind of algorithms and methods that are used to make these systems to be able to distinguish fraudulent behaviour from non-fraudulent. The later part of this thesis describes the implementation and testing, one being a supervised in form of a ANN, the other being a unsupervised in form of k-means clustering with outlier detection. The prototype is mainly coded in Erlang, consisting of the three subparts, the two mentioned classifiers (ANN and clustering) and a web-GUI.

The reason for choosing to implement an ANN is that it is easily customized for different domains along with its great potential of finding patterns that are difficult for humans to see. The customizability was a crucial factor since data from the domain that the prototype is meant for was not available from the start of the thesis. The reason for choosing k-means clustering along with outlier detection is that unsupervised methods have a great potential for classification and are not as extensively used in fraud detection as supervised methods.

During testing we saw that the ANN was able to distinguish fraudulent behaviour from non-fraudulent with an accuracy of 70-90% depending on what kind of setup and training sets that is used. The reason that the accuracy spans from 70-90% is for several reasons. One reason is that different neuron setups were tested. For the data used in the evaluation we saw that the setup 41-50-2 was the most efficient. Another reason is that the ANN does not handle symbolic data. It only handles continuous variables, resulting in that we mapped all symbolics to numeric values. The ANN then uses these values for computation and sometimes these symbolic values could be for ex-

ample the protocol used for communication, for example 'HTTP'. Another flaw in the SUSPECT prototype is that the supervised classifier is synchronous resulting in that training takes a long time, since one has to wait for one set to pass both ways before being able to send in the next. The values of the neurons are also just stored in volatile memory, resulting in that if one neuron crashes, the whole system needs to be re-trained.

During the testing of we also saw that the supervised classifier had better accuracy than the unsupervised classifier. The unsupervised classifier had an accuracy around 50-60% as seen in table 5.4.

The unsupervised classifier was tested with the same data as the supervised. Unsupervised classification might work better when the data compared is not from a global source but rather compares user behaviour to the same user's past behaviour. This would work better since new clusters should look somewhat similar to the past ones, which is not true for a global distribution since how different users and services behave differs a lot.

Comparing the SUSPECT prototype with other implementations that have done classification of the data mentioned in section 5.1 is hard. Since SUSPECT does store the weights of the neurons in volatile memory, a small error in a test run can cause a failure that might result in that the training needs to be repeated. As mentioned earlier SUSPECT does not handle symbolic data and assigns every symbolic entry a numeric representation. IP numbers are a part of the test data as well and during some test runs these causes an arithmetic overflow and/or underflow that results instability and errors in SUSPECT. Due to this SUSPECT has not yet done a successful run on the whole test data. This results in that a comparison between other implementation and itself is hard to do since the quantity of data used in the other systems differs too much in size compared to the amount used in SUSPECT.

Future work

Due to the time constraints of this thesis, SUSPECT is only a prototype, it needs polish before being ready for extensive use.

Future work on the SUSPECT prototype would consist of choosing a better clustering algorithm. Another thing would be to use test data that does not contain global entries, but rather user-profile data to compare a user's behaviour to the same users past behaviour, thus making outliers more noticeable.

Right now SUSPECT only supports continuous values but would perform better if it could handle symbolic values as well. This would need to be adjusted in how the neurons computation methods work.

The ANN is coded to be parallelized for faster results but right now all layers need to wait for the previous which results in the parallelization being close to none. Altering how this works would speed up the training process significantly.

Summary and Conclusions

Fraud detection is a field which can never rest, fraudsters do exist and always will. This game of cat and mouse will never end. In this thesis several algorithms for fraud detection are presented and two of them implemented and tested.

Supervised algorithms and methods showed good performance in finding fraudsters in larger data sets, the downside being that they have to be taught what is considered fraudulent and not. Unsupervised algorithms on the other hand seems good in theory, but the prototype implementation did not perform well. A few reasons why this is so is the following: the cluster centers were adjusted continuously making all the entries always being close to a center. The distribution of the data was not so spread resulting in smaller tight clusters. Another approach for choosing centers would have been in its place. The Euclidean distance was used as a measurement, choosing another metric would probably increase performance drastically.

Appendices



Used technologies

A.1 Erlang

Erlang is a functional, concurrent, fault-tolerant language. It first appearance was in 1986 and was back then a language owned by Ericsson but was open-sourced in 1998. Development in Erlang is swift thanks to its functional nature and is one of the reason that the prototype developed in this thesis is coded in Erlang. Another reason for choosing Erlang as the language for the prototype is that the system that the prototype is supposed to interact with is mainly coded in Erlang.¹

A.1.1 Cowboy

An open-source HTTP server coded in Erlang. The HTTP server part of the prototype is using cowboy to serve HTML pages.²

A.1.2 Erlydtl

An open-source library that converts DTL to Erlang bytecode making it possible to use DTL along with Cowboy.³

¹<http://www.Erlang.org/>

²<https://github.com/extend/cowboy>

³<https://github.com/evanmiller/erlydtl>



The KDDCup'99 test data and suspects altering of it

The data consists of CSV files, where each row contains 42 values, some elements are continuous values and some symbolic.

The items are the following:

1. duration: continuous.
2. protocol_type: symbolic.
3. service: symbolic.
4. flag: symbolic.
5. src_bytes: continuous.
6. dst_bytes: continuous.
7. land: symbolic.
8. wrong_fragment: continuous.
9. urgent: continuous.
10. hot: continuous.
11. num_failed_logins: continuous.
12. logged_in: symbolic.
13. num_compromised: continuous.

14. root_shell: continuous.
15. su_attempted: continuous.
16. num_root: continuous.
17. num_file_creations: continuous.
18. num_shells: continuous.
19. num_access_files: continuous.
20. num_outbound_cmds: continuous.
21. is_host_login: symbolic.
22. is_guest_login: symbolic.
23. count: continuous.
24. srv_count: continuous.
25. serror_rate: continuous.
26. srv_serror_rate: continuous.
27. rerror_rate: continuous.
28. srv_rerror_rate: continuous.
29. same_srv_rate: continuous.
30. diff_srv_rate: continuous.
31. srv_diff_host_rate: continuous.
32. dst_host_count: continuous.
33. dst_host_srv_count: continuous.
34. dst_host_same_srv_rate: continuous.
35. dst_host_diff_srv_rate: continuous.
36. dst_host_same_src_port_rate: continuous.
37. dst_host_srv_diff_host_rate: continuous.
38. dst_host_serror_rate: continuous.

39. dst_host_srv_error_rate: continuous.

40. dst_host_error_rate: continuous.

41. dst_host_srv_error_rate: continuous.

42. type: symbolic.

The symbolic data which are numbers are left untouched. Symbolic data that are strings are converted into numbers. Each string in each group are assigned a unique integer for that group in the order of appearance starting at 1.

Bibliography

- [1] Kdd cup 1999. <http://www.kdd.org/kdd-cup-1999-computer-network-intrusion-detection>. Accessed: 2013-08-06.
- [2] Results of the kdd'99 classifier learning contest. <http://cseweb.ucsd.edu/users/elkan/clresults.html>. Accessed: 2013-08-06.
- [3] Richard J. Bolton and David J. Hand. Unsupervised profiling methods for fraud detection. *Proc. Credit Scoring and Credit Control VII*, pages 5–7, 2001.
- [4] Richard J. Bolton, David J. Hand, and David J. H. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–249, 2002.
- [5] Leo Breiman. *Classification and Regression Trees*. Wadsworth Publishing Co Inc, 1984.
- [6] Vladimir Estivill-Castro. Why so many clustering algorithms: a position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75, June 2002.
- [7] K. Fanning, K.O. Cogger, and R. Srivastava. Detection of management fraud: a neural network approach. In *Artificial Intelligence for Applications, 1995. Proceedings., 11th Conference on*, pages 220–223, Feb 1995.
- [8] S. Ghosh and D.L. Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630, Jan 1994.

- [9] D.J. Hand. *Discrimination and classification*. Wiley series in probability and mathematical statistics: Applied probability and statistics. J. Wiley, 1981.
- [10] M.H. Hasson. *Fundamentals of artificial neural networks*. Bradford books. Mit Press, 1995.
- [11] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605 vol.1, 1989.
- [12] Constantinos S Hilas and John N Sahalos. User profiling for fraud detection in telecommunication networks. In *5th International Conference on Technology and Automation*, pages 382–387, 2005.
- [13] M. Kamber J. Han. *Data Mining - Concepts and Techniques*. Diane Cerra, second edition, 2006.
- [14] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, Jul 2002.
- [15] Yufeng Kou, Chang-Tien Lu, S. Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 2, pages 749–754 Vol.2, 2004.
- [16] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *Networking, sensing and control, 2004 IEEE international conference on*, volume 2, pages 749–754. IEEE, 2004.
- [17] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate analysis*. Probability and mathematical statistics. Academic Press, 1979.
- [18] G.J. McLachlan. *Discriminant analysis and statistical pattern recognition*. Wiley Series in Probability and Statistics. Wiley, 1992.
- [19] K. Mehrotra, C.K. Mohan, and S. Ranka. *Elements of artificial neural networks*. Complex Adaptive Systems Series. MIT Press ; London, 1997.
- [20] M. I. Petrovskiy. Outlier detection algorithms in data mining systems. *Programming and Computer Software*, 29(4):228–237, 2003.

- [21] Jon T.S. Quah and M. Sriganesh. Real-time credit card fraud detection using computational intelligence. *Expert Systems with Applications*, 35(4):1721 – 1732, 2008.
- [22] J.R. Quinlan. *C4.5: Programs for Machine Learning*. C4.5 - programs for machine learning / J. Ross Quinlan. Morgan Kaufmann Publishers, 1993.
- [23] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, May 2000.
- [24] Silvia Schiaffino and Anala Amandi. Intelligent user profiling. In Max Bramer, editor, *Artificial Intelligence An International Perspective*, volume 5640 of *Lecture Notes in Computer Science*, pages 193–216. Springer Berlin Heidelberg, 2009.
- [25] S. Viaene, R.A. Derrig, and G. Dedene. A case study of applying boosting naive bayes to claim fraud diagnosis. *Knowledge and Data Engineering, IEEE Transactions on*, 16(5):612–620, May 2004.