An Investigation Of the Role Of Feature Selection On the Classification Performance Of Machine Learning Algorithms

Tamer Aldwairi
Distributed Analytics and Security Institute
High Performance Computing Collaboratory
Mississippi State University
Starkville, MS 39762, USA
Department of Mathematics
and Computer Science
Ursinus College
601 E. Main Street
Collegeville, PA 19426, USA
taa70email@gmail.com

Dilina Perera
Distributed Analytics and Security Institute
High Performance Computing Collaboratory

Mark A. Novotny
Department of Physics and Astronomy
Texas A&M University
College Station, TX 77843-4242, USA
dilinanp@tamu.edu

Abstract

The recent and rapid increase in the size of data and the large variety of data types that are required to be processed within different scientific fields has made it extremely important to find an optimal set of features that reduces the complexity of the data and extract valuable and beneficial information from within these large sums of unorganized data. This is especially true in the cases where datasets suffer from the problem of high dimensionality. These challenges require an in-depth understanding of the effect of feature selection algorithms and how to utilize them to achieve higher rates of accuracy within high dimensional data sets. In this paper we investigate the role of proper feature selection in the context of classification of anomaly network intrusion detection systems. We show that the appropriate selection of important features can have a huge effect on the accuracy level of the classifiers. We also compare the performance of different classifiers, and show that the Restricted Boltzmann Machine classifier outperforms other type of classifiers in some cases.

1. Introduction

Feature selection is considered one of the main vital steps and building blocks of the data pre-processing approach. Recently more light has been shed on its importance due to the sudden growth in the scale of data that needs to be processed within the web. The accelerated surge in the speed of the Internet allowed people to exchange, upload and download data at higher unprecedented speeds than ever before. This, coupled with an increase in the size and the dimensionality of data has led to a rise in the complexity of the data.

This sudden growth in the data size, its various types and the speed required to process such data came to be known as “Big Data”. Big Data has had its vast effect on almost every scientific field were data is gathered and generated in large quantities. This data does not follow the same rules like our conventional data, and cannot be analysed using the same assumptions we are used to. Even though the Big Data problem has been addressed by a number of different researchers, a large portion of them concentrate on the problem from the perspective of the size or volume of the data leaving aside the problem of huge increase in the dimensionality of the data. The increase in data dimensionality in some cases could be even more important than the problem of the fast increase in the volume of the data, especially when the number of features increases to more than one million features, a case known as feature explosion [1].

That is why certain techniques like feature selection stands out as a solution that paves the way to reduce the dimensionality and the complexity of the data, with minimum loss of information that is considered crucial to achieve our required objective of a faster anomaly detection along with higher accuracy, sensitivity and specificity. When building an anomaly network intrusion
detection systems (A-NIDS) the task is to build a predictive model (i.e. a classifier) that can identify attacks through classifying network traffic behaviour as normal or anomalous [2].

2. Methods

2.1 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) is a probabilistic generative stochastic artificial neural network that consists of two main layers, one of visible units and the other of hidden units. The visible units represent the observable data and the hidden units represent the relationships between those data units. The visible and hidden units are connected to each other through edges, where each edge has a weight associated with it. In the restricted form of the Boltzmann machine, which we are using in this paper, there are no direct connections between units of the same type. Basically there are no edges between nodes of the same layer [3, 4].

The joint probability distribution in a binary RBM is given by the Gibbs distribution

\[ p(v, h) = \frac{1}{Z} e^{-E(v, h)}, \]

where \( Z = \sum_{v, h} e^{-E(v, h)} \) is the partition function,

with its energy function given by

\[ E(v, h; \theta) = -\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j. \]

The \( m \) (\( n \)) visible (hidden) units are \( v \) (\( h \)).

The RBM was originally introduced in the 80’s, but did not receive a lot of attention due to the limitation in the computational power of computers at that day and age. With the recent increase in computational power and the advancement of high performance computing, Geoffrey Hinton as well as other researchers in the field of machine learning and pattern recognition [5-12] showed that a RBM is able to learn complex patterns of data with high dimensional features. In addition to they showed that RBMs offers high levels of accuracy, and has a great potential in distinguishing different visual and auditory patterns.

2.2. The NSL-KDD Dataset

KDDCup99 is a set of simulated network traffic data in a military environment [13]. The KDD dataset consists of approximately 4.9 million single connection vectors, each vector contains 41 features. The KDD data set has been used for a long time as a benchmark to evaluate intrusion detection systems.

The KDD data set suffers from some inherent weaknesses and drawbacks [14]. To solve those intrinsic problems the NSL KDD dataset was created based on the KDD data set. The NSL-KDD does not solve all the problems associated with the KDD data set, but nevertheless it solves a significant number of those problems. NSL-KDD still suffers from some issues [15], but it is still considered a more effective set to benchmark and compare intrusion detection systems than many of its counterparts.

The advantages of using NSL-KDD over KDD can be summarized in the following points:

• Redundant records were removed from the training set to omit biases of the classifiers towards those records.

• Duplicated records are not included in the test set to remove the biases of the learners caused by methods with higher detection rates on frequent records.

• The number of records in the NSL-KDD is chosen by sampling a higher percentage of records for each group with higher prediction difficulty level – the groups are categorized into different levels based on how hard it is for the learners to correctly predict the correct class (normal/anomaly) the data set falls under – and are a lower percentage of the total records in the original data set. This approach allows more accurate evaluation when applying different learning techniques on the data.

• The number of records is reasonably small compared to the original dataset to enable one to run the experiments on the whole dataset, without the need to select a smaller subset of the dataset.

2.3 Data Set Pre-processing

Since we are aiming to train the data sets using one of the machine learning algorithms (Contrastive Divergence, Persistent Contrastive Divergence), we need to convert the network traffic data into a binary form suitable for training the machine learning algorithms. It is important for training purposes that when we convert the data into its binary representation, we need make sure that the size of the resulting vectors for each data set maintains the same order as the original data to be able to train the data effectively.

To produce valid and reliable models we need good preparation of the data. Pre-processing the data requires usually all or some of these steps:
Data cleaning: - Two invalid records were found and removed from the KDD99 data when creating the NSL-KDD data set.

Data integration: - There is no need for data integration for this dataset, since we are not using multiple datasets from different resources.

Data transformation: - Sometimes normalization or standardization is needed and can make the task of extracting important data easier on the classifiers.

Data reduction: - Choosing the NSL-KDD saves us from using the data reduction step. Data in the NSL-KDD was reduced from 4.9 million records to 125,973 in the training set, and from 311,029 to 22,544 in the test set.

Data discretization: - Discretization simplifies and speeds up the process of learning [16]. The challenge in the discretization technique is to find the cut point which divides the range into a set of separate intervals with minimum loss of information, and where those intervals are coherent [17]. Discretization of continuous data is very important, as failure to do so can lead to a flaw in the analysis of the data. Different classifier algorithms have different sensitivities regarding the extent in which they are affected by discretization. Nevertheless correct usage of discretization can simplify the task of extracting useful information from the data.

The first step in the discretization process is finding out the different types of data used by each feature, and which ones needs to be discretized. We have 41 features, out of them 34 have real value ranges, 4 are binary (i.e. 0 or 1) and 3 are nominal excluding the last feature (i.e. feature number 42) which is either normal or anomalous. Regarding the binary and the nominal features, no discretization of the data is needed. In fact the binary data does not need any additional data pre-processing.

The conversion of the nominal data can be done for each feature by enumerating the number of different instances within each feature, and mapping each distinct instance of the data set to a certain unique binary value. For example, take the protocol type feature which contains three instances TCP, UDP and ICMP. Then for each time we encounter the TCP instance in our data set it will be mapped to the binary equivalent of 00, and for UDP it will be mapped to 01, and for ICMP it will be mapped to 10. We performed a similar procedure on all the nominal features within that dataset. Table 1 demonstrates the conversion of the data instances for the three different nominal features of the NSL-KDD.

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Binary Value</th>
<th>Service</th>
<th>Binary Value</th>
<th>Flag</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>0</td>
<td>Aol</td>
<td>0</td>
<td>OTH</td>
<td>0</td>
</tr>
<tr>
<td>UDP</td>
<td>1</td>
<td>Auth</td>
<td>1</td>
<td>REJ</td>
<td>1</td>
</tr>
<tr>
<td>ICMP</td>
<td>10</td>
<td>Bgp</td>
<td>10</td>
<td>RSTO</td>
<td>10</td>
</tr>
</tbody>
</table>
|               | Couri
|               | 11           | RSTO    | 11           |
|               | csnet_nes    | 100     | RSTR         | 100  |
|               | Ctf           | 101     | S0           | 101  |

The conversion of the real values is the most challenging task. Direct conversion of real values to binary can prove to be challenging. The reason for that boils down to the fact that decimal point representation requires more storage space (i.e. more bits per each value) than normal integer values. In addition, the decimal point increases the complexity of the data that the algorithm is trying to learn. If there are many instances with decimal values, then this will increase the size of each record which will lead to increasing the computational complexity required by the model to learn each record, and will slow the algorithm’s performance.

To overcome this problem, we need to discretize the data to convert the continuous data (i.e. real values) into a form of distinct data ranges. Discretization of the data can be done in many different ways and using a variety of approaches. This makes the discretization process a non-trivial task that requires a careful understanding of the dataset characteristics in order to choose the most effective discretization approach. Our purpose here is to discretize the data with minimal loss of information. For our data, we choose to discretize the real values in a supervised fashion with respect to their normal and anomaly classes. After discretizing the data, the ranges are converted into an equivalent integer representation and then into binary values.

For example, let us consider the feature called “count” (which represents the number of connections to the same destination host as the current connection in the past two seconds), which is the twenty third feature of the KDD data set. The numeric values of this feature range from 0 to 511. Discretizing this feature in a supervised manner with respect to the normal and anomaly class will result in having 28 ranges for the real values, usually called bins.
Figure 1. The WEKA output for discretization of the “count” feature into 28 bins.

We then take each bin and assign to it a numeric integer value, usually in the same order as the real values within the range, and then convert it to its binary equivalent. The order of the integer values should be preserved to allow meaningful comparisons.

Table 2. The conversion of the ranges into integer values and then into binary.

<table>
<thead>
<tr>
<th>Feature Name (Count)</th>
<th>Integer Value</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-inf-1.5]</td>
<td>0</td>
<td>00000000</td>
</tr>
<tr>
<td>(1.5-2.5]</td>
<td>1</td>
<td>00000001</td>
</tr>
<tr>
<td>(2.5-5.5]</td>
<td>2</td>
<td>00000010</td>
</tr>
<tr>
<td>(5.5-10.5]</td>
<td>3</td>
<td>00000111</td>
</tr>
<tr>
<td>(10.5-13.5]</td>
<td>4</td>
<td>00001000</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>(502.5-inf)</td>
<td>27</td>
<td>00011011</td>
</tr>
</tbody>
</table>

3. Evaluation

To evaluate the performance of RBM against other machine learning algorithms, we need to train the RBM using the Contrastive divergence [18] or the Persistent contrastive divergence [19] algorithm, and tune its hyper-parameters to find optimal or semi optimal settings for the RBM. Once the RBM is trained using a certain collection of hyper-parameters, we validate its performance on a separate test set to get an assessment of the effectiveness of our training procedure. The hyper-parameters that were tuned during the training were the learning rate (0.005, 0.001, 0.05 and 0.01), the number of iterations (500, 1000, 5000, 10000), the batch size (500, 1000, 5000, 10000), and the k-value (1, 10).

4. Results

To evaluate the effectiveness of our RBM training, we compared the performance of RBM to other state of the art algorithms used for classification in machine learning. Those algorithms are J48 (a Java implementation of the popular C4.5 algorithm [20]), Naïve Bayes [21], NB Tree, Random Forest [22], Random Tree, Multi-Layer Perception (MLP), and Support Vector Machine (SVM) [23, 24].

To be able to get a valid comparison between the performances of the different algorithms, we trained and tested all seven algorithms using the same datasets used before or during the RBM training and testing. Figure 2, 3, and 4 demonstrate the performance of our RBM against all seven algorithms using three different datasets. It can be clearly seen that the RBM’s performance is comparable to other state of the art machine learning algorithms when used for classification of network traffic datasets.
all features were included in the comparison our RBM outperformed all other classifiers. When the number of classifiers was reduced to 22 and 12 features, we found that the performance of the RBM was on par with the other classifiers. We believe that further investigation is needed to reach a solid conclusion regarding the effect of feature selection on machine learning classifiers. Future work on this aspect could be comparing the effect of other feature selection algorithms on the performance of different classifiers.

5. Conclusion

In this paper we showed the effect of proper feature selection on the performance of different machine learning classifiers for a network intrusion detection system. In most cases, the selection of the features will result in an improvement or a slight decrease in the performance of the classifiers. We also evaluated the performance of a RBM against other known classifiers that are are commonly used for network traffic classification tasks, and found that when

References


