Project Report

11/7/16

1 Topic Title

Intrusion Detection

2 Team Members

Daniel Silva
Jacob Tarnow
Simon Kwong

3 Timeline

Week	Date	Task Description	Results
1	10/02 - 10/08	Develop Project Proposal	Completed
2	10/09 - 10/15	Data Processing	Utilizing 10% of dataset
3	10/16 - 10/22	Feature Selection	31/41 predictors featured
4	10/23 - 10/29	Feature Creation	Determined target variable
5	10/30 - 11/05	Status Review | Create Models | Prediction Algorithms	LDA, QDA, LDR, etc.
6	11/06 - 11/12	Run Models | A/B Testing	
7	11/13 - 11/19	QA Models | Prune Data | Update Models	
8	11/20 - 11/26	Plot All Comparisons | Determine How To Portray Results	
9	11/27 - 12/03	Clean Up Any Errors | Write Up Project Challenges/Results	
10	12/04 - 12/10	Final Presentation	

4 Problem Definition

Adapted from: http://kdd.ics.uci.edu/databases/kddcup99/task.html
The main problem is to find a way to easily distinguish normal network connections from network intrusions/attacks. We will approach the learning task via supervised learning where we will build a classification model to help solve this problem. The data holds various parameters that have already been labeled as: TCP, HTTP, normal, bad, etc., which we will be using for training. Then we will test on this data. The target variable, Y, which we will have to create, will be to predict whether a specific connection is good or bad. Y can take on different values that discern it from normal or bad, especially in respect to the types of attacks.

4.1 Definition

Software to detect network intrusions and protect a computer network from unauthorized users and insiders. This intrusion detector learning task is to build a predictive classification model capable of distinguishing between intrusions or attacks and normal connections.

4.2 Setting

In 1998 the DARPA Intrusion Detection Evaluation Program surveyed and evaluated intrusion detection simulated in a military network environment. Because the majority of people use the internet, it is important to prevent the user's privacy and personal information from leaking out and to prevent unauthorized users from gaining access to this information. Within the last decade, many instances of cyber-attacks have been documented. One of the more notable attacks was an attack against a Ukraine power grid. This caused more than 200,000 residents and local businesses to experience a blackout, essentially costing banks and businesses to lose millions

4.3 Type of Learning

We will approach network intrusion detection with supervised learning, more specifically, classification. In our data set, each tcp data packet falls into five categories: a normal connection, a denial-of-service (DOS), an unauthorized access from a remote machine (R2L), an unauthorized access to local superuser or root privileges (U2R), and probing or scanning. We can classify these categories into two main categories of either a "good" normal connection, or a "bad" intrusion connection. For every new tcp connection coming in, we want to predict and classify whether or not the network access is an intrusion or a normal connection. It is also important to note that there are specific attacks distributed in the test data set, but not in the training data. Because some intrusion experts believe that most novel attacks are variants of known attacks, having unknown attacks is more realistic

5 Data Analysis

The dataset for this project has been supplied via KDD Cup 1999 Data Information and Computer Science, University of California, Irvine. Last modified on October 28, 1999. The raw training data is about 4GB of compressed binary. The TCP dump data contains seven weeks of network traffic. The data was processed into roughly five million connection records.
Source: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

5.1 Features

Gathered from http://kdd.ics.uci.edu/databases/kddcup99/task.html, the features outlined by Stolfo, defined features that help in distinguishing normal connections from bad connection, i.e. attacks. They categorized the features into the following: same host, same service, time-based traffic, host-based traffic, and content features. Same host features examine only connections in the past two seconds. These features have the same destination host as the current connections. Same service features examine connections that have the same service as the current connection in the past two seconds. Both of these together, same host and same service, are defined as time-based traffic. Host-based traffic on the other hand, involves sorting connection records by destination host. Thus, focusing on the same host instead of a specific time window. Finally, content features are added features that help in determining other predictors that may add to certain behaviors in the data.

5.2 Feature Analysis

Provided in the KDD Cup Dataset, there are a total of 41 features, 10 of which are undocumented. In order to determine whether or not a connection is a good or bad connection, every feature listed below plays a role in resolving the incoming access type.
P-Value Significance is denoted by (***) in the table below.

Feature	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 to different hosts	continuous

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Continuous = Quantitative; Discrete = Qualitative/Classifier

5.3 Statistical | Variable Importance

After running both the generalized logistic regression model and quadratic discriminant analysis model, all features proved statistically significant except the attribute for *num_outbound_cmds*. This made sense as all the observations for this one predictor were 0's.

6 Models | Algorithms

Due to the fact that our target variable is a class with k > 2 possible values based on the network connections and types of attacks, we are considering the following models: Logistic Regression, QDA (quadratic discriminant analysis), and SVM (anomaly detection). In order to have a relatively good model for this, SVM, Bagging, and Random Forests are the best options. An alternative is to over-sample all of the bootstrapping, or to use no random sampling, with the method call of stratified. The use of bagging and random forest would benefit us as we have a large amount of data and a large overhead of computation. We will need to create a specific sample size for our model testing, thus stratifying our sampling brings many benefits. The benefits of stratifying the sampling is that it allows us to determine the best stratification for our data, yet still ensuring the minimum sample necessary to satisfy our constraints (network connection of good/bad).

7 Prediction Accuracy

For our Logistic Regression Model we were able to achieve 80% accuracy.

```
> mean(glm.pred == train$access_type)
[1] 0.7986134
```

For our Linear Discriminant Analysis we were able to achieve 96% accuracy.

```
> mean(lda.pred$class == train$connection_type)
[1] 0.964884
```

8 Iterations

For each of the models that we proposed above, we completed five iterations per model. We believe five iterations per model is sufficient due to the amount data and time complexity. The below code outlines our iterations per model. The number of iterations completed were 25.

```
> proc.time() - glm.fit.time
   user  system elapsed
235.947  10.100 247.574

> proc.time() - lda.time
   user  system elapsed
 27.312   2.944  30.290
```

9 Challenges

The main challenge is the amount of time spent for computation. Our current dataset is a reduced number of rows, roughly 10% from the original dataset of 400K records. If we work with a smaller dataset we could possibly get relatively more accurate predictions, yet we still need to measure skewness. When we find skewness we can then try to balance our data more appropriately.