

Kuan Li

Mar 27, 2018

## **Report of Knowledge Discovery and Data Mining (KDD)**

### **Abstract**

This work presents a data preprocessing and variate control experiments to support data mining and predict network attack types. This paper starts with an analysis of the KDD Cup 99 data sets[1] and their potential for machine learning study. The data set consists of connection records with 41 features whose relevance for intrusion detection are not clear. All traffic is either classified “normal” or specific attack types which could classified into the four attack types denial-of-service(DOS), network probe (PROBE), remote-to-local (R2L) or user-to-root (U2R). By analyzing the characteristic of KDD data set, this paper lies emphasis primarily on two parts: First part is discovering general nature of KDD data set and suggesting the settings which could be applied to any similar data set. Second part is prediction test which aims to show accuracy in different prediction types and evaluate the efficiency of the model.

### **1.Introduction**

KDD is a typical data set used for network secure study and Intrusion Detector Learning. The data set is consisted with server log information which has over 4,000,000 instances and exactly 42 attributes. As for the training tool, this paper chooses NeurophStudio which is a lightweight Java neural network framework to develop common neural network architectures. Due to this software, we will use “Total mean square test error” which is the mean square of output minus desired output instead of the accuracy in most of the test. This is because in this paper, we will observe the tendency in the most experiments, test error is directly show in NeurophStudio and is enough to show the tendency of changes. The relationship between max error and accuracy is the lower the test error is, the higher the accuracy is.

Based on KDD and NeurophStudio, this paper covers the data processing, parameter setting experiment and prediction test. All of them are essential tasks for machine learning. Data processing aims to normalize data which could accelerate convergence and improve the accuracy of the trained model; Parameter setting experiment aims to find the proper training parameters which could maximize the efficiency and accuracy of the model; Prediction test aims to show accuracy in different prediction types and evaluate the efficiency of the model. All of the processes mentioned in this paper are provided as suggestions for readers working on KDD and any other similar data sets.

## **2. Problem Statement**

First of all, the data set itself need to go through complicated normalization process. From the perspective of data processing, there are attributes described as “words” such as the attack-types or protocol types. How do we convert these words? From the perspective of normalization, we faced even harder problems which are extremely huge numbers. For example, some users connect to the server for only a few seconds while some users will connect up to  $10^8$  seconds to the server. How do we deal with these huge difference? Or what kind of normalization algorithm should we use? Both of them are challenges that we need to find a proper way to solve because the data set greatly decides the training outcome.

From the perspective of finding the proper experiment settings, what are the proper parameter settings for the training max error, learning rate or hidden neurons for KDD data set? How they influence the training process? What is the relationship between each parameters? Additionally, there is also a challenging problem which is the outcomes fluctuation for each “same” experiments. For example, although the two experiments are sat with same parameters, the outcomes of two experiments are usually different because every time at the beginning, the model will randomly initialize the weights. So how do we make the statistics process more accurately?

Prediction is one of the most important part in machine learning. The most critical parameter here is “accuracy” which shows the model is good or not. Since we can only get max error in NeuropgStudio, how do we get accuracy? And there are also many practical questions like what is the difference between predicting the specific attacks and attack groups like dos or u2l? Is the difference huge? We now are predicting the existed attack types, what if we predict the attacks that we have never seen? Could our model still achieved high accuracy? In this work, we will discuss all of these questions.

### **3. Data Prepossessing**

Before setup the experiment, we need to process the data set first. There are three primary processing steps which are text processing, normalization and sampling.

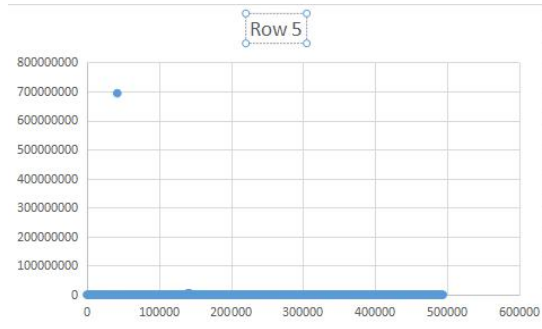
#### **3.1 Text processing**

First, text processing. This paper used 10% data set of KDD and found the row 2(protocol type), 3(service), 4(flag), 42(attack type) are words that can't be learned by machine. So we went through text processing to convert words into numbers. We wrote several C++ programs to first show the word attributes and then converted them into numbers uniformly into the scale of 0-1. For example, in the row 42, there are 23 different attack types. In order to make them fall into 0-1, they are divided into consecutive numbers with the distance 0.045. For instance, '0' represents 'normal', 0.045 represents 'buffer\_overflow', 0.09 represents 'loadmodule'.

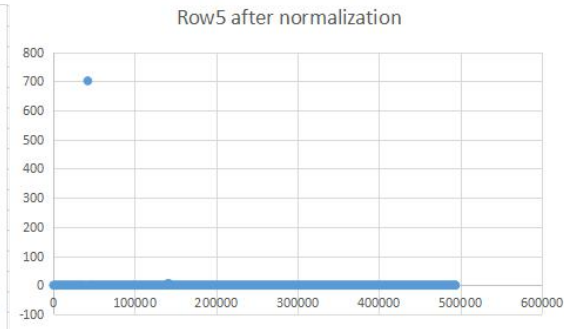
#### **3.2 Normalization**

Second step, normalization. We found in the some rows, some numbers are extremely large while others are small. For example, in row 5 and 6, the ranges are  $0-6.9 \times 10^8$  and  $0-5.1 \times 10^6$ . To improve the speed and efficiency of convergence, we planed to do normalization in row 5(src\_bytes), 6(dst\_bytes), 23(count), 24(srv\_count), 32(dst\_host\_count), 33(dst\_host\_srv\_count). In row 23 and 24, the ranges are 0-511.

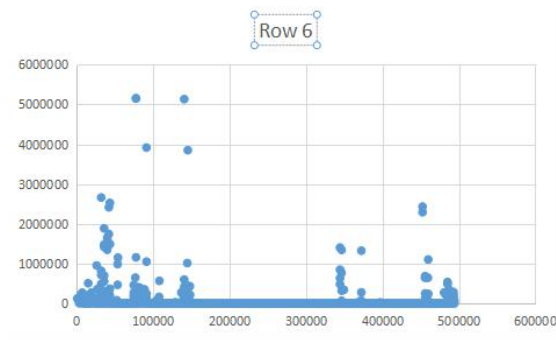
In row 32 and 33, the ranges are 0-255. So we planned to do min-max algorithm to these 4 rows. But for row 5 and 6, we needed to change the algorithms because of the abnormal max value. So we opened excel to first observe the data distribution of row 5 and 6.



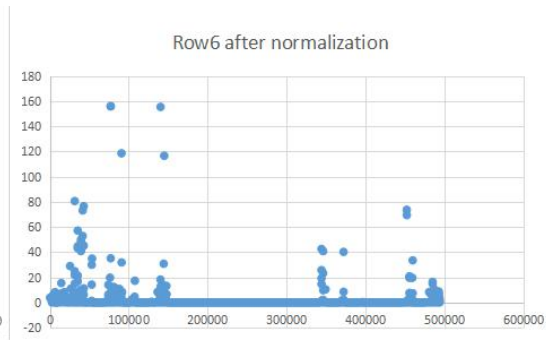
The distribution in row 5



The distribution after normalization



The distribution in row 6



The distribution after normalization

As we can see, in row 5, the difference between most of the values and abnormal values are super huge. To narrow this difference and put the values in to a rational scale, I planed to use z-score algorithm which is do  $x' = \frac{x - \mu}{\sigma}$  for each pieces of data.

The data after normalization obeys the distribution which the mean value is 0 and Variance is 1. After the normalization, row 5 lies in the scale of -0.003 - 700 and row 6 lies in the scale of -0.02 - 160. It greatly narrows the range to 100,000. Although 100,000 is also a big number for normalization, but it is much smaller than 100,000,000 which is the range of original data set. This algorithm also retains the characteristic of original data and in later test, and this algorithm also had advantages than min-max or log method in achieving smaller test error.

### **3.3 Sampling**

Last step, sampling. The number of 10% KDD Data set is about 490,000 which is still a huge number for learning or testing. So we then chose 1% of KDD Data. And we use random sampling to make sure the 1% data could represent 10% of KDD Data. The random sampling module is actually generating a random number between 0-490,00 and choose the pieces of data corresponding to this number. We do it 49,000 times and we got 49,00 pieces of data. Last, in this 49,000 pieces of data, we use random sampling again to chose 80% of learning data and 20% of testing data.

## **4. Experiment Design & Setup**

Then, we got normalized 40000 training data and 10000 test data. Each piece of data has 42 attributes. In these 42 attributes, 41 are used as input attributes which are server connection log and 1 is the output which is the specific attack type. Next, from the perspective of experiment design, it is divided into 2 parts which is find the proper experiment settings and prediction test. First experiment is consisted with 5 groups of experiments which are training result between hidden neurons, max error, learning rate and improvement of cutting attributes. Second experiment is consisted with 2 experiments which are all attacks vs attack groups, predicting new attack types. Each experiments will be trained with sigmod algorithm.

### **4.1 Find the Proper Experiment Settings**

For the standard of finding proper experiment settings, accuracy comes first. If the accuracy could achieve best under the tolerable iteration times. We regard the parameter as the proper experiment settings.

#### **4.1.1 Hidden Neurons Experiment**

As we know, hidden neurons is a crucial factor that influences the converge speed. And the more hidden neurons, the more training times it will cost. Here comes the

question, how does hidden neurons influence KDD? What is the proper hidden neuron number for KDD? We will solve these questions in this experiment.

To make sure the fluctuation of each single test is controlled in a limit of range, I did each test twice and record the mean data of two tests. The first experiment will show the relationship between hidden neurons and test error. The second experiment will explore the relationship between hidden neurons and iteration times. In both experiments, the max training error will be set 0.001 because it will avoid converging too quickly, learning rate is default (0.2), momentum is default(0.7).

#### **4.1.2 Training Max error Experiment**

This test is to find a proper max error for training this data set. As we all know, with the decreasing of training max error, the test error will also decrease based on the common sense. But is that real? If it's real, what kind of tendency lies in them? What is the proper training max error for this test?

Here also did 2 experiments which are the accuracy and time experiments. First is the relationship between training error and test error. Second is the training error between iteration times. The hidden neurons is set as 24 because it will produce more accurate result and learning rate is default.

#### **4.1.3 Learning Rate Experiment**

Learning rate determines how much an updating step influence the current value of weights. With the increasing of learning rate, the weight will change more quickly. So we want to find a learning rate that is low enough that the network converges to something useful, but high enough that we don't have to spend too much time. So this experiment is between learning rate and test error and the hidden neurons is 24, max training error is 0.001.

#### **4.1.4 Improvement: Cut off attributes**

There is total 42 attribute in the data set. But how many of them are really useful in prediction? In other words, how many of them are crucial for making the prediction?

So it is necessary to know if we could cut some useless attributes to improve the processing speed and decrease the data noise. Here is the process I go through.

First, I found the row 20, 21 are always 0, so I deleted these two rows. Then, I made histogram for each attributes and tried to found something. But I found even “dst\_host\_same\_srv\_rate” and “dst\_host\_diff\_srv\_rate” don’t not match to each others which means have the different impacts on final result.

Then I found a paper[2] which tells how to reduce unnecessary attributes. This paper introduces a crucial method using decision tree which is testing the result of cutting one attribute and look how it changes the test error. If this cut attribute has low information gain and doesn’t have huge impact on the test error, it will be regarded as unnecessary attribute and cut off. Based on the outcomes of that paper, I cut off all the unnecessary attributes and reserve the 11 important attributes which are duration, protocol\_type, service, src\_bytes, dst\_bytes, wrong\_fragment, error\_rate, dst\_host\_srv\_count, dst\_host\_diff\_srv\_rate, dst\_host\_same\_src\_port\_rate, dst\_host\_error\_rate. And I tested this 11 attributes of data with the 6 hidden neurons ( because the total attributes are cut by 1/4, hidden neurons are also cut by 1/4 ), 0.2 learning rate and 0.001 max error.

## **4.2 Prediction Test**

This part is consisted with 3 experiments, prediction of attack groups vs all attacks, new types prediction.

### **4.2.1 Attack Groups & All attacks**

As we mentioned before, we did text convert for all attacks in text processing. So the outcomes will fall into a specific attack types. This is the way we use in training and test. But we can’t ignore another way to do this is classified each attacks into attack groups which are dos, u2r, u2l and probe. Attack groups test is an important way to show the performance of our trained models.

Here comes the experiment with 4 attack groups and 1 group of all attacks. Each attack groups containing all normal traffic, but only one out of the four attack groups. To make sure there is enough test instances for each attack groups, different sampling scale should be used in different groups. For DOS attack, because this type is really common, we couldn't use all of the 10% KDD data. So this group is extracted from the test data we used in the previous experiment. For U2R, U2L and PROBE attack, because they are rare cases, so these are extracted from 10% KDD data to make sure there are enough test instances. Then we used the same trained model to test the accuracy and compared the accuracy for each test groups.

#### **4.2.2 New Attack Types Prediction**

In the real world, server will sometimes encounter new attack types that we have never seen before. So here comes the practical problems, can our model detect them as attack types?

In this experiment, we use KDD test data which includes about ten new attack types. Then we extracted only these new attack types and see how our model will react. But there comes other questions. First, how do we define new attack types? Because our aim is to know "it is an attack", we didn't generate new types for them, instead, we regarded them as existed attacks but we don't care the exact attack types, just regarded them as "abnormal".

Second, how do we calculate the accuracy? Our method is using the difference between output and desired output of each pieces of data. If the difference is less than half of the difference between each specific attack types or normal type, it could be successfully distributed to the correct type. So we regarded them as "right". In detailed, in the output row, '0' represents 'normal' while other numbers represent specific attack types and the distance between each types are 0.045. So in the predicting experiment, if the predicted number '>' 0.0225, it means the piece of data could be classified into attacks, so it means the predication success. We use that way to know the number of 'right' predication. Then we count the "right" data pieces and divided by total number of data. So we get the accuracy!



Last, in this experiment, we calculated the accuracy of existed attack types and new attack types and compared them. The test data of existed types comes from former 10000 pieces of KDD data. The new attack types data comes was extracted from KDD test data and the data after extracting only includes new attack types. The number of new attack types is 18729.

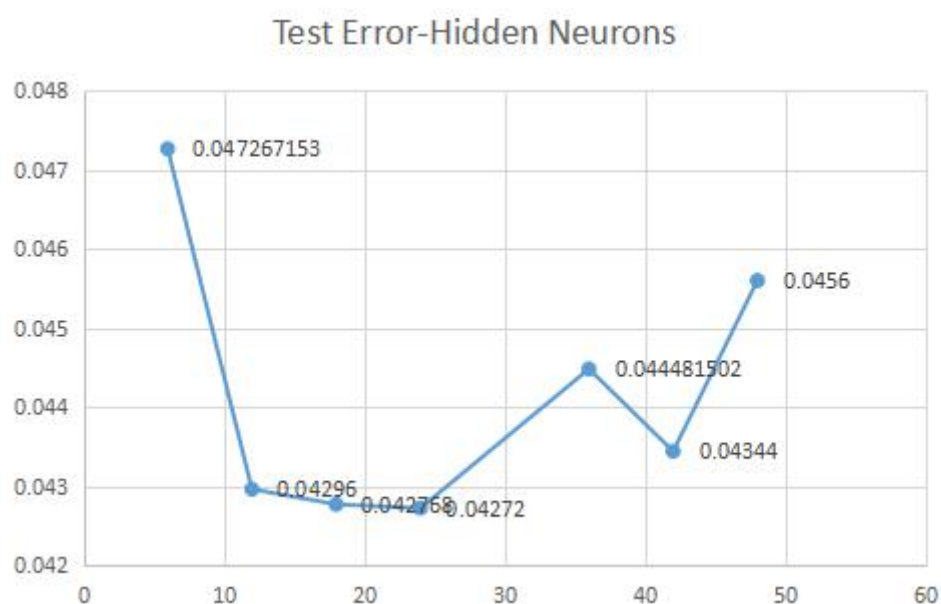
## 5. Data Analysis & Explanation

Based on the experiment design and setup, here comes the experiment data and the explanation of the data.

### 5.1 Find the Proper Experiment Settings

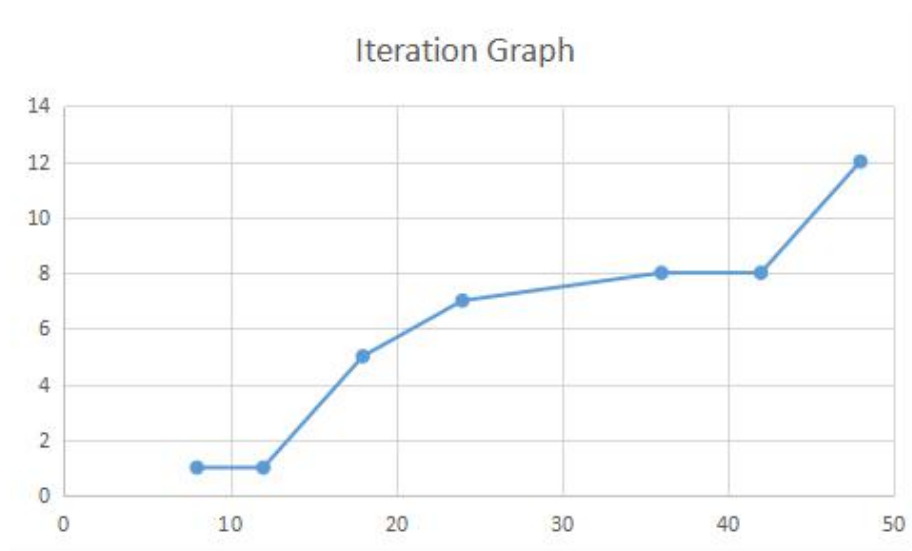
#### 5.1.1 Hidden Neurons Experiment

Here is the first experiment, We tested 8, 12, 18, 24, 36, 42, 48 hidden neurons to get this graph. the training iteration times from each hidden neurons are 1, 1, 5, 7, 8, 8, 12. Blow is the error graph between hidden neurons and test error, X-coordinate is the number of hidden neurons, Y-coordinate is test error. The lower the test error is , the higher accuracy it has. And the hidden neurons are selected 8, 12, 18, 24, 36, 42, 48.



as we can see, when the hidden neurons is between 18-24, the learning effect is good. And when the hidden neurons is 24, it becomes the best. But actually the difference is not so huge that I can say 24 is the best because every tests will have a fluctuation. So I would say that I suggest to set 12 hidden neurons because the model has good precision and cost shorter time. Additionally, we can see that when hidden neurons=12, the precision is small enough and precision doesn't change too much later. So we can make an assumption that the essential attributes are about 12 because when each essential attributes got a responded hidden neurons, the precision will reach a small enough range.

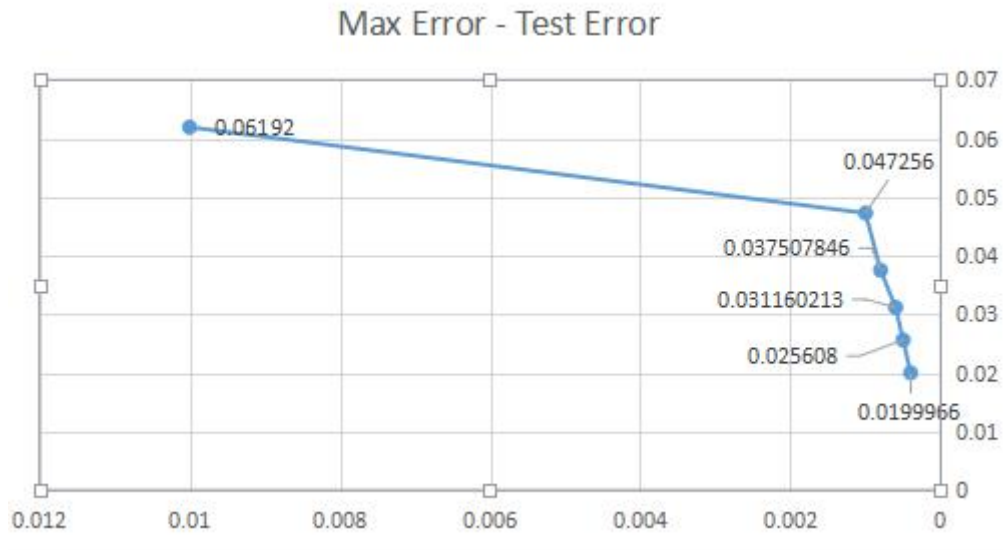
Blow is the iteration graph, X-coordinate is the number of hidden neurons, Y-coordinate is iteration times.



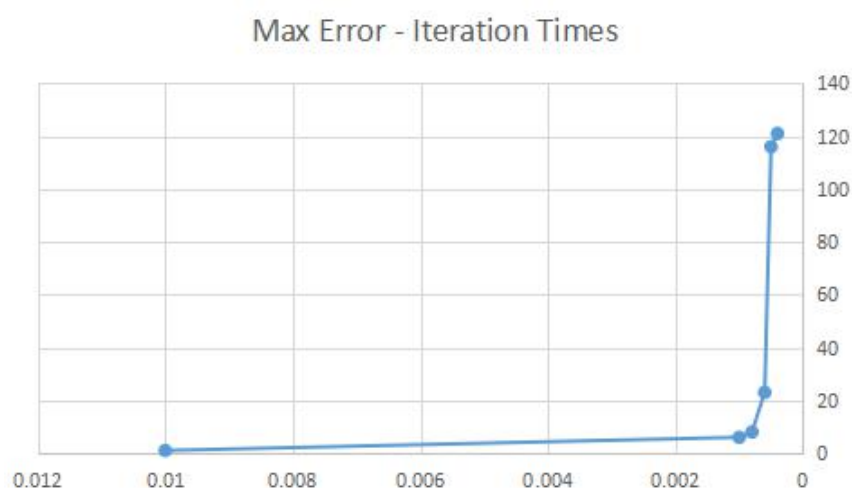
It's obvious that more hidden neurons need more iteration times and will cost more time. But actually the training time is not very different due to the several iteration times. So the iteration times is tolerable in this experiment.

### 5.1.2 Training Max error Experiment

Here shows the relationship between max error and test error first. I sat max error 0.01, 0.001, 0.0008, 0.0006, 0.0005, 0.0004 and got the test error curve blow. X-coordinate is the Training Max Error, Y-coordinate is test error.



As we can see, when max training error is decreasing from 0.001, 0.0008, 0.0006, 0.0005, 0.0004, the test error decreases from 0.06 to 0.02. And we could see their relationship is liner relationship. So here comes the question, is max error=0.0004 the best point for the test? I think the answer is yes, but when the max error is lower than 0.0004, the system will take much more time to train and the iteration times is not tolerable. Here comes the graph between max error and iteration times. X-coordinate is the Training Max Errors which are 0.01, 0.001, 0.0008, 0.0006, 0.0005, 0.0004, Y-coordinate is iteration times.

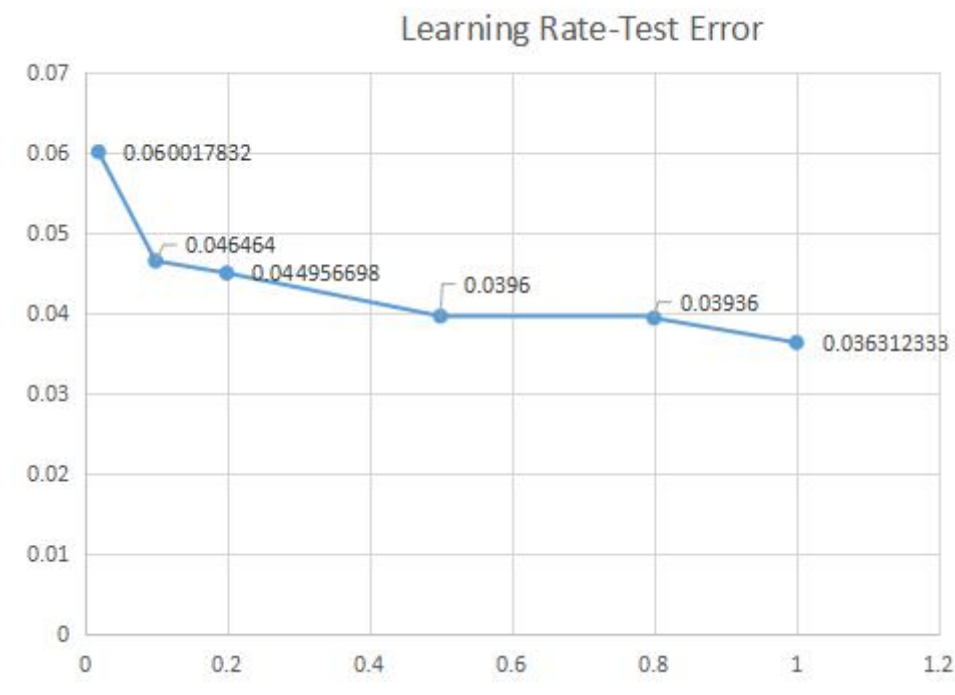


As we can see, it is an exponent curve which means a much more iteration times cost. '0.004' also has tolerable iteration times. But when exceed 0.004, the iteration times

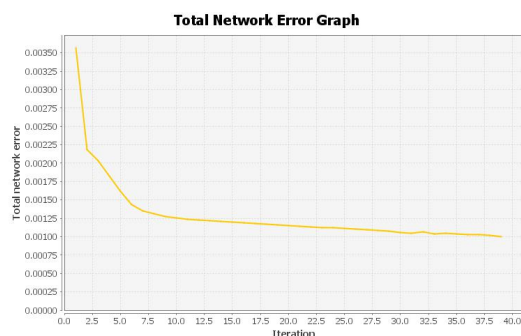
will be so many that can't be tolerable.

### 5.1.3 Learning Rate Experiment

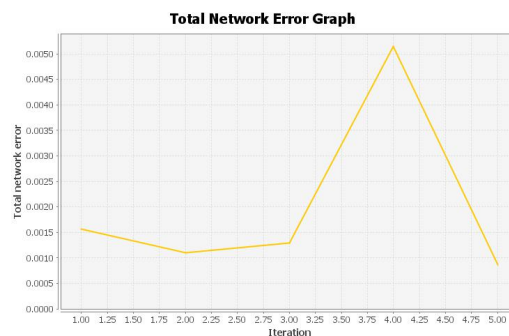
Here is the graph between learning rate and test error. X-coordinate is the Learning Rate which is 0.02, 0.1, 0.2, 0.5, 0.8, Y-coordinate is test error.



In the error graph, I found that with the increasing of learning rate, the tendency of test error is decreasing. And one thing I found interesting is that although low learning rate costs much more time and has lower accuracy, its curve is more smooth and stable. We can see in the graph when learning rate=0.5, there is a “hill” which means great fluctuation.



Learning Rate=0.02 Smooth Curve

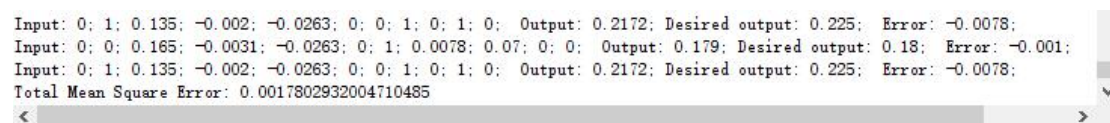


Learning Rate=0.5 “hill” curve

Although the hill curve does exist, It is not very common. I just encountered 1 time among 10 times test. But it does matter the training model because it means the training model is not very stable. Actually the precision differences of learning rate after 0.2 are not so huge, so I prefer to set learning rate to 0.2 which is the default parameter with low error rate and higher stability.

#### 5.1.4 Improvement: Cut off attributes

Here is the training curve: X-coordinate is iteration time, Y-coordinate is the test error.



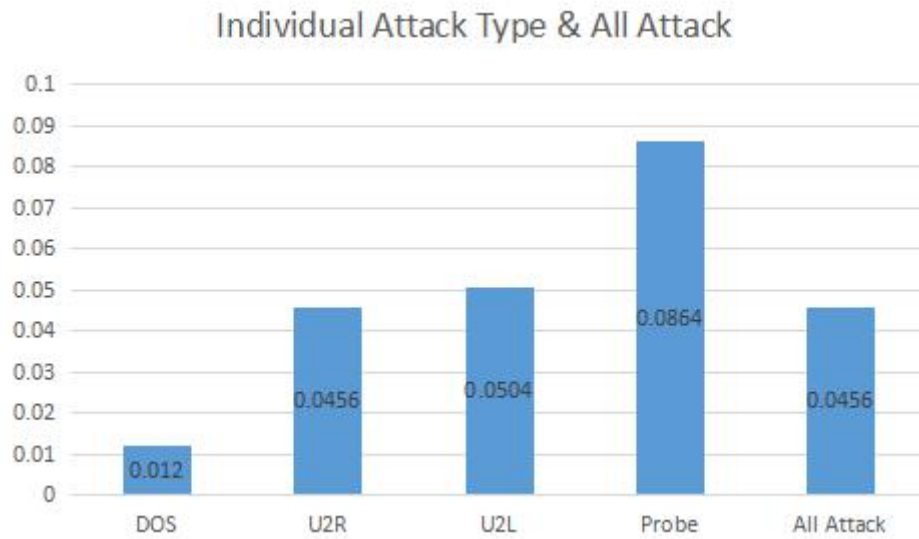
Input: 0; 1; 0.135; -0.002; -0.0263; 0; 0; 1; 0; 1; 0; Output: 0.2172; Desired output: 0.225; Error: -0.0078;  
Input: 0; 0; 0.165; -0.0031; -0.0263; 0; 1; 0.0078; 0.07; 0; 0; Output: 0.179; Desired output: 0.18; Error: -0.001;  
Input: 0; 1; 0.135; -0.002; -0.0263; 0; 0; 1; 0; 1; 0; Output: 0.2172; Desired output: 0.225; Error: -0.0078;  
Total Mean Square Error: 0.0017802932004710485

As we can see, the perfect curve shows the perfect convergence. The error is 0.00178 which is similar or even better to my previous test 0.0019 under the same parameters. This method is very magical because even I don't know the specific domain knowledge, I can also cut off the "noise" and retain important attributes. Furthermore, recalling the hidden neurons test, the 11 attributes also correspond with the lower accuracy change after 12 hidden neurons which means the important attributes are about 12.

## 5.2 Prediction Test

### 5.2.1 Attack Groups& All attacks

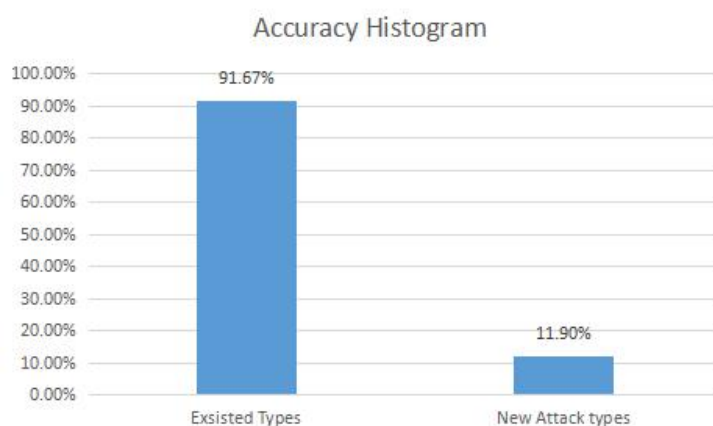
Here is the graph for attack groups and all attacks. X-coordinate is the test types, Y-coordinate is the test error.



As we can see in this graph, dos attack has the lowest error which means highest accuracy, probe attack has the highest error which means lowest accuracy. U2R, U2L and all attacks has the similar error index which means similar accuracy. It happened to produce this formula:  $\text{Accuracy (all attacks)} = \frac{\text{Accuracy (DOS + U2R + U2L + PROBE)}}{4}$ . It coordinate with the data it self because all attacks include these 4 types of attacks. So it proves that our model is stable and the accuracy is real accuracy which does make sense.

### 5.2.2 New Attack Types Prediction

Here is the accuracy graph between existed types and new attack types:



The accuracy of predicting existed types is 91.67% while the accuracy of predicting new attack types is only 11.90%. Why the accuracy of predicting new types is so low compared to existed types? On the one hand, it is because the new types of data only

includes new attack types, it doesn't include predicting normal types. Normal types is indeed a large scale of data and could "attenuation" data set. On the other hand, it shows that our model is not good at predicting new types of attacks.

## **6. Conclusion**

First, This paper investigates KDD data set and discovers the desired settings which could be applied to any similar data set. Desired hidden neurons are 12 because 12 after 12 hidden neurons, the training error doesn't change too much but has faster convergence speed and the result 12 important inputs in cutting attributes experiment also correspond with the 12 hidden neurons. Desired training error is 0.004 because the model will be more precise under a tolerable iteration times. Desired learning rate is 0.2 which is because the low error rate and high stability.

Second, The investigate of predicting test. The accuracy of predicting all attacks equals to one quarter of the accuracy in predicting DOS, U2R, U2L and PROBE attacks. The accuracy of predicting new attack types is about 12% which is not likely to predict new attack types. While the accuracy of predicting existing specific attack types is about 92% which is a fairly good accuracy.

## **7. Reference**

- [1] KDD Cup 1999 Data, [http : //archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data](http://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data)
- [2] Ralf C. Staudemeyer, Christian W. Omlin, "Extracting salient features for network intrusion detection using machine learning methods", SACJ 52, July 2014