A REPORT ON

NETWORK INTRUSION DETECTION SYSTEM USING FUZZY LOGIC

[In partial fulfillment of the course:
*Data Mining **CS F415**]*

Submitted by:

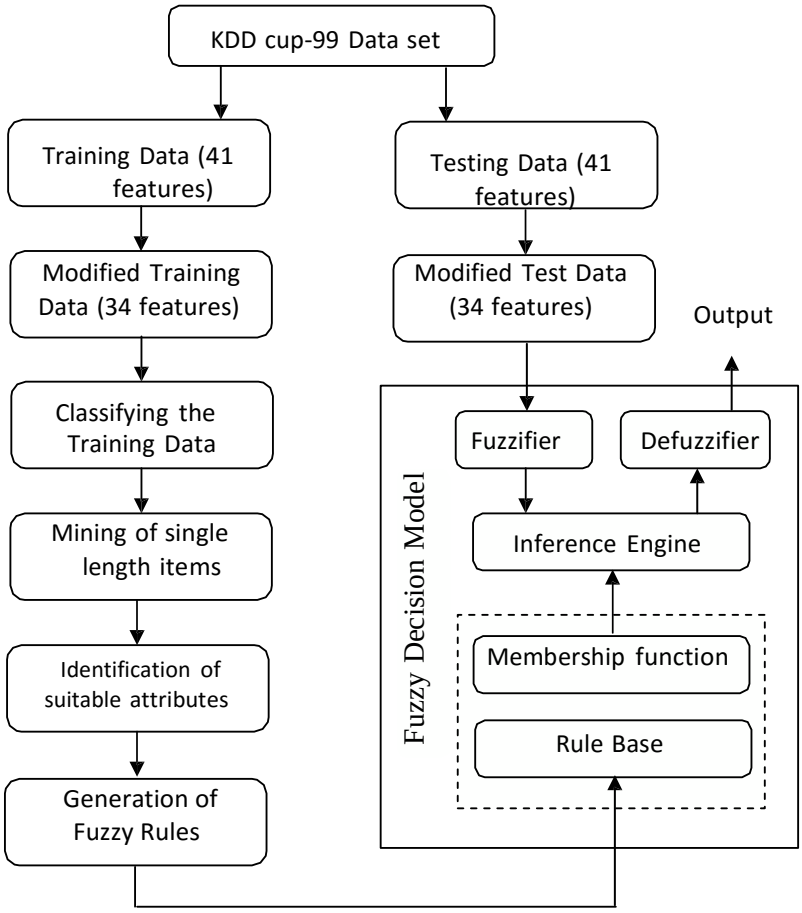
Sanuj Bhatia	**2013A3PS230P**
Anish Bansal	**2013A8PS171P**

Project Group Number: 13

Instructor-in-Charge: Mr. Yashvardhan Sharma
(April 2016)

NETWORK INTRUSION DETECTION SYSTEM USING FUZZY LOGIC

Presently, it is unfeasible for several computer systems to affirm security to network intrusions with computers increasingly getting connected to public accessible networks (e.g., the Internet). In view of the fact that there is no ideal solution to avoid intrusions from event, it is very significant to detect them at the initial moment of happening and take necessary actions for reducing the likely damage. One approach to handle suspicious behaviors inside a network is an intrusion detection system (IDS). For intrusion detection, a wide variety of techniques have been applied specifically, data mining techniques, artificial intelligence technique and soft computing techniques. Most of the data mining techniques like association rule mining, clustering and classification have been applied on intrusion detection, where classification and pattern mining is an important technique. Similar way, AI techniques such as decision trees, neural networks and fuzzy logic are applied for detecting suspicious activities in a network, in which fuzzy based system provides significant advantages over other AI techniques. We have used Fuzzy Logic for Intrusion Detection System



The above flowchart is our exact approach that we have worked upon in this project. Our data set was KDD99 Cup Data Set which had 41 attributes in it. We reduced the number to 34 by removing all types but continuous because the algorithm we are going to use for data mining can deal only with continuous data. Once, we have this Modified Training Data Set, we have to classify it into normal data or attack data.

Mining of single length frequent items

At first, frequent items (attributes) are discovered from both classes of input data and by using these frequent items, the significant attributes are identified for the input KDD-cup 99 dataset. In general, frequent item set are mined using various conventional mining algorithms, such as Apriori. These algorithms are suitable to mine frequent item set with varying length only for the binary database, which contains only the binary values. But, the input dataset (KDD cup-99) contains continuous variable for each attributes so that, the conventional algorithm is not suitable for mining frequent items. By considering this property, we simply find the 1-length items from each attributes by finding the frequency of the continuous variable present in each attribute and then, the frequent items are discovered by inputting the minimum support. These frequent items are identified for both class namely, normal and attack.

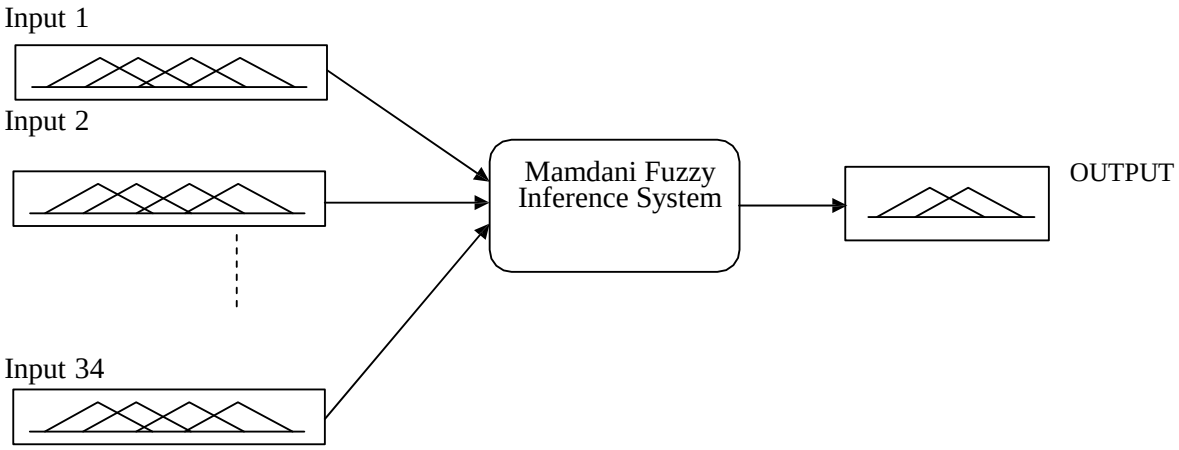
Identification of suitable attributes for rule generation

To identify suitable attributes, we follow a very simple technique. As we have the whole data classified as attack or normal data, therefore, we will check the maximum and minimum for all 34 attributes. Assume the case of Attribute 1, if the maximum and minimum of this attribute is same for both attack and normal data, it will be insignificant to our analysis and if they are different, they are significant to us.

Rule generation

The effective attributes chosen from the previous step is utilized to generate rules that is derived from the {max, min} deviation. By comparing the deviation range of effective attributes in between the normal and attack data, the intersection points are identified for the effective attributes. By making use of these two intersection points, the *definite* and *indefinite rules* are generated. For example, {max, min} deviation range for normal data related to attribute1 is {1, 5} and {max, min} deviation for attack data corresponding to attribute1 is {2, 8}. Then, the rule is designed like, *"IF attribute1 is greater than 5, THEN the data is attack*, *"IF attribute1 is in between 2 and 5, THEN the data is normal OR attack"* and *"IF attribute1 is less than 2, THEN the data is normal"*. In addition to that, some of the data contains only one intersection point, which provides only two rules.

There was one trend that was very common in our data i.e. one of the attack or the normal data's range was the subset of the other. Whichever had the larger range was given more weight for the range exclusive to it and the other data was given more weight in the common range.
This is more clearly explained in the Fuzzy Rules we have made on the Fuzzy Toolbox.
The Final Fuzzy Diagram is shown below.



Finding an appropriate classification for a test input

For testing phase, a test data from the KDD-cup 99 dataset is given to the designed fuzzy logic system for finding the fuzzy score. At first, the test input data containing 13 attributes is applied to *fuzzifier*, which converts 13 attributes (numerical variable) into linguistic variable using the triangular membership function. The output of the fuzzifier is fed to the *inference engine* which in turn compares that particular input with the rule base. *Rule base* is a knowledge base which contains a set of rules obtained from the *definite rules*. The output of inference engine is one of the linguistic values from the following set {Low and High} and then, it is converted by the *defuzzifier* as crisp values. The crisp value obtained from the fuzzy inference engine is varied in between 0 to 1, where '0' denotes that the data is completely normal and '1' specifies the completely attacked data.

Code

```
clear all;
%Loading the KD cup 99 dataset. The original data set comprises of 41
%attributes which was reduced to 34 after removing the symbolic attributes.
load('proj_data.mat');
%Initializing a matrix with all values equal to zero
cnt=zeros(1,34);
% APRIORI ALGORITHM ; Mining frequent 1-length item sets - evaluating the
% frequency of continuous variables of each attribute (Modifies Binary
% Apriori Algorithm for continuous features)
for i=1:34
    for j=1:211791
        if(project1(j,i)~=0)
            cnt(1,i)=cnt(1,i)+1;
        end
    end
end
% setting support equal to 100. We identified the attributes which affected
% only one of the five possibilities and removed them as they had no impact
% on classification.
tot=0;
for i=1:34
    if(cnt(1,i)>=100)
        tot=tot+1;
        dataAfterApriori(:,tot)=project1(:,i);
    end
end
%tot = 23; 11 attributes were found insignificant
normalData=dataAfterApriori(1:98468,:);
attackData=dataAfterApriori(98469:211791,:);
%Finding range of each attribute for normal and attack data and if the
%range came out equal, we removed the corrosponding attribute.
rangeNormal=max(normalData)-min(normalData);
rangeAttack=max(attackData)-min(attackData);

isRangeEqual=zeros(1,tot);
for i=1:tot
    if(rangeNormal(1,i)==rangeAttack(1,i))
        isRangeEqual(1,i)=1;
    end
end

tot2=0;
for i=1:tot
    if(isRangeEqual(1,i)==0)
        tot2=tot2+1;
        dataWithSignificantFeatures(:,tot2)=dataAfterApriori(:,i);
    end
end
% tot2 = 14; By the end of the pre-processing step, only 14 significant
% attributes remain

% Defining the 15th column by the output column of the initial dataset
dataWithSignificantFeatures(:,tot2+1)=project1(:,35);
```

```
shuffledArray =
dataWithSignificantFeatures(randperm(size(dataWithSignificantFeatures,1)),:);

%splitting the complete dataset into training and testing datasets in the
%ratio 70:30
training=shuffledArray(1:ceil(0.7*211791),:);
testing=shuffledArray(1+ceil(0.7*211791):211791,:);

%In order to have reproducible results, training testing data once produced
%was stored and loaded for the next part of the code, i.e., Rule Generation
load('training.mat');
training = sortrows(training,tot2+1);

normalData=training(1:68871,1:14);
attackData=training(68872:148254,1:14);

%From this step, we used the Fuzzy toolbox for Fuzzy rule generation and
%filtering. Deviation matrices were used to find the intersection which
%helped in generation of rules.
deviationNormal=[min(normalData);max(normalData)];
deviationAttack=[min(attackData);max(attackData)];

% Fuzzy Rules were fed into the Fuzzy toolbox and an FIS file was generated.
% It will then be used to evaluate the Fuzzy System
% Loading the test data to evaluate the fuzzy system developed
load('TestFuzzy.mat');
% Actual Output is extracted from the original dataset
actualOut=testFuzzy(:,13);
% Output Column removed from the Testing Data
testFuzzy=testFuzzy(:,1:12);
% Loading the FIS file(Fuzzy System)
fismat=readfis('Fuzzy.fis');
% Evaluating the system via the test data
out=evalfis(testFuzzy,fismat);

% Initializing the count to zero
tp=0;
tn=0;
fp=0;
fn=0;
evalOut=zeros(63537,1);
for i=1:63537
% Finding evalOut by comparing the output obtained from the Fuzzy System
% with the set threshold value
    if(out(i,1)>0.8)
        evalOut(i,1)=1;
    end
% Comparing the evaluated output with the actual Output
    if(evalOut(i,1)==actualOut(i,1))
        tp=tp+1;
        if(evalOut(i,1)==0)
            tn=tn+1;
            tp=tp-1;
        end
    end
    if(evalOut(i,1)~=actualOut(i,1))
        fp=fp+1;
```

```
        if(evalOut(i,1)==0)
            fn=fn+1;
            fp=fp-1;
        end
    end
end
% Obtaining the precision, recall, f-measure and the accuracy of the attack
% as well as the normal data for both training and testing data sets.
Aprecision=tp/(tp+fp);
Arecall=tp/(tp+fn);
beta=1;
AfMeasure=((beta^2)+1)*Aprecision*Arecall/(((beta^2)*Aprecision)+Arecall);

Nprecision=tn/(tn+fn);
Nrecall=tn/(tn+fp);
beta=1;
NfMeasure=((beta^2)+1)*Nprecision*Nrecall/(((beta^2)*Nprecision)+Nrecall);

accuracy=(tp+tn)/(tp+tn+fp+fn);
```

Results

		Training	Testing
	Precision	0.9082	0.9082
Normal	Recall	1	1
	fMeasure	0.9519	0.9519
	Accuracy	0.953	0.9529
	Precision	1	1
Attack	Recall	0.9123	0.9118
	fMeasure	0.9541	0.9539
	Accuracy	0.953	0.9529

Precision (tp/tp+fp) is the fraction of retrieved instances that are relevant, while recall (tp/tp+fn) is the fraction of relevant instances that are retrieved. The F-Measure is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the harmonic mean, i.e. F-Score = 2(Precision × Recall)/(Precision + Recall)

As can be seen, the results are satisfactory both for normal and attack data, and are almost equal for training and testing data, since the data is large and training using fuzzy of such a varied set cannot result in overfitting.

Bibliography

1) www.ijcse.com/docs/IJCSE11-02-01-034.pdf
2) Dr. Fengmin Gong, “Deciphering Detection Techniques: Part II Anomaly-Based Intrusion Detection”, White Paper from McAfee Network Security Technologies Group, 2003.
3) J. Allen, A. Christie, and W. Fithen, “State Of the Practice of Intrusion Detection Technologies”, Technical Report,CMU/SEI-99-TR-028, 2000.
4) ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4530019
5) Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu and Ali A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", in Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications, pp. 53-58, Ottawa, Ontario, Canada, 2009.
6) Bharanidharan Shanmugam, Norbik Bashah Idris, "Improved Intrusion Detection System Using Fuzzy Logic for Detecting Anamoly and Misuse Type of Attacks", in Proceedings of the International Conference of Soft Computing and Pattern Recognition, pp: 212-217, 2009.
7) Zadeh, L.A., “Fuzzy sets”, Information and control, vol.8, pp. 338-353, 1965.