4-21-2006

Unsupervised Learning to Improve Anomaly Detection

Daniel H. Garrette '06
Illinois Wesleyan University

Unsupervised Learning to Improve Anomaly Detection

Daniel H. Garrette
Department of Mathematics and Computer Science
Illinois Wesleyan University
Bloomington, IL
dgarrett@iwu.edu

April 21, 2006

Abstract

An intrusion detection system (IDS) is used to determine when a computer or computer network is under attack. Most contemporary IDSs operate by defining what an intrusion looks like and checking traffic for matching patterns in network traffic. This approach has unavoidable limitations including the inability to detect novel attacks and the maintenance of a rule bank that must grow with every new intrusion discovered. An anomaly detection scheme attempts to define what is normal so that abnormal traffic can be distinguished from it. This thesis explores the ways that an unsupervised technique called "clustering" can be used to distinguish normal traffic from anomalous traffic. This thesis will also explore an attempt to improve upon existing clustering algorithms to improve anomaly detection by adding in limited amounts of a posteriori knowledge.

1 Introduction

The purpose of an intrusion detection system (IDS) is to determine when a computer or computer network is under attack. Intrusion detection systems take many forms. The form most common in commercial systems is that of a rule-based system. These systems detect intrusions by matching network traffic patterns against a bank of patterns for known intrusions. Though this technique has been the most successful to date, it has certain unavoidable limitations. First, a rule-based system relies solely on its bank of known intrusions. This means that it will never

be able to detect a new intrusion until that intrusion has been discovered and documented as a rule in the rule bank. Secondly, the rule bank must grow with every new intrusion discovered. This means that the bank of rules will at some point become large enough to cause problems with storage and ability to be searched. Thirdly, these rules must be written by IDS experts, meaning that end users cannot be fully independent. They must continually receive new rules from the experts.

These problems can be overcome by anomaly detection techniques. This approach seeks to determine what is "out of the ordinary" and to mark it as "intrusive". Because of the huge amounts of network traffic that exist on any network, it is an extremely difficult task to label traffic as either normal or anomalous. Therefore many experts see a need for anomaly detection techniques that do not require labeled data. One approach to accomplishing this goal is to build clusters of network traffic. This works by dumping huge amounts of network traffic into a grid and letting a computer group the data. It can then be inferred that the data in larger groups constitute normal traffic and the data in smaller groups constitute anomalous traffic.

The goal of this research was to explore this form of anomaly detection. A variety of established clustering algorithms will be implemented and tested. In addition, an attempt was made to improve the accuracy of these algorithms through modifications in the implementations.

2 Unsupervised Learning

There are two general approaches to machine learning. The first, and most common, is called "supervised learning." It comprises the categories of machine learning algorithms in which an "instructor" is involved. The "instructor" in a supervised learning algorithm is the set

of correct labels for all examples. For unsupervised learning, no such labels exist and, therefore, learning must be done independently of such advance knowledge.

It is obvious that a supervised learning algorithm will yield better results than an unsupervised one as more information will lead to a more accurate classification function. One might be tempted, because of this fact, to try to use supervised learning techniques for intrusion detection. However, this may not be the best approach. The data used for a machine learning technique for intrusion detection is network traffic. A supervised learning algorithm requires labeled data, but because a network experiences such huge amounts of traffic, it would be impossible for any organization implementing one such IDS to have intrusion detection experts label all of that data so that it could be used for learning. Therefore, it makes much more sense to develop unsupervised learning techniques that can be applied to the data in its natural form.

One form of unsupervised learning that seemed to lend itself well to the problem of intrusion detection is called "clustering." A clustering algorithm is one that takes a large number of individual data pieces and groups them based on similarity. The next few sections will discuss the different clustering algorithms that were explored and implemented throughout this research project.

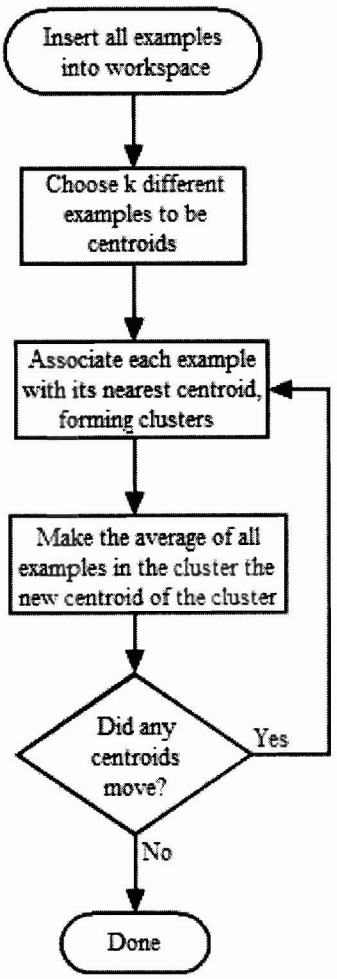

Figure 1
K-Means Algorithm

2.1 K-Means Clustering

The K-Means clustering algorithm [1] is a popular unsupervised learning technique. Given an integer constant *k*, and a group of data, the algorithm will partition the data into *k* distinct groups based on the

similarities and differences of the data examples. The algorithm is given in Figure 1.

Despite its excellent accuracy, the K-means algorithm suffers from two major problems that make it not a suitable candidate to act as the clustering agent in an intrusion detection system. The first is due to the parameter k that must be passed so that the algorithm knows how many clusters to make. There does not seem to be any way to automatically generate the value of k prior to running the algorithm. It may be possible for an expert to determine what k should be before running the clustering algorithm, but this is clearly something that should be avoided since it is our goal to minimize the amount of expert knowledge required to prepare and run the intrusion detection system. Because of this problem, a few variations of K-Means, such as Y-means [4], which is described in the next section, have emerged that do not require a predefined value for k.

The second major problem that dooms the K-Means algorithm's viability as a tool for clustering in an intrusion detection system is its time complexity. The algorithm passes over the entire list of examples in the data set an indeterminable number of times. Despite the fact that the algorithm is guaranteed to converge, it may take an extremely long time for this to occur. This is a problem inherent to the nature of the K-Means algorithm and it cannot be overcome through slight modifications.

2.1.1 The Y-Means Algorithm

One of the variations of K-Means that was created to overcome the problem of dependence on a predefined value for the number of clusters is the Y-Means algorithm [4], developed by Guan et al. The Y-Means algorithm begins by choosing a random value for k that is between one and the number of instances in the data set. It then begins an iterative sequence

like that of K-Means but with the added steps of splitting clusters that are too large and merging clusters that overlap. In doing this, it is hoped that the algorithm will converge on the ideal number of clusters, thus having generated the value of k without having known it ahead of time.

During this research, the Y-Means algorithm was implemented and tested it as the clustering component of an intrusion detection system. The problem that arose in the implementation, however, was that unless the data had clearly defined clusters, groups of data that are very similar and very distinct from all other data, the algorithm would not converge. The problem was in the process of splitting and merging. If there were no clearly defined clusters in one area of space, then an endless loop of splitting and merging of the same data points would begin. When tests were run on the implementation, this problem was encountered and no test runs using the Y-Means algorithm ever completed.

2.1.2 K-Means with Splitting

In order to have a variation of K-Means that is not dependent on a predefined value for the number of clusters but that was not subject to the infinite loop problem of Y-Means, the implementation was changed to not perform any merging of clusters. The flow chart of this modified algorithm is shown in Figure 2.

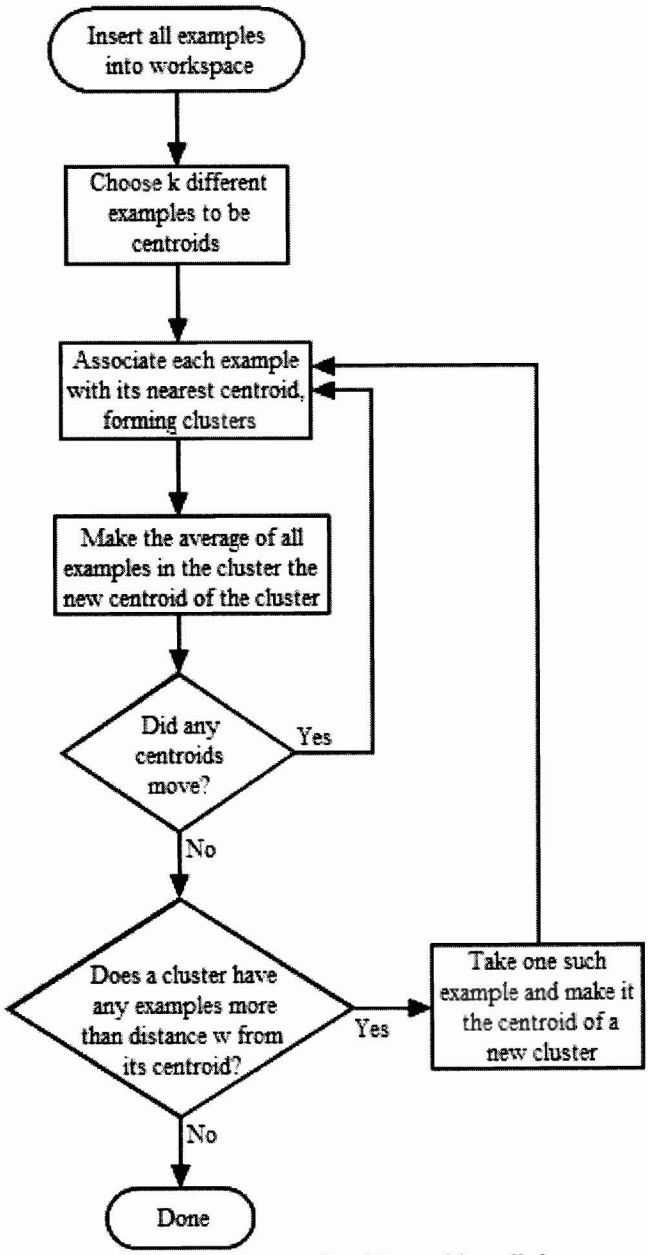

Figure 2: K-Means Algorithm with Splitting

It quickly became obvious in testing that this was a much better approach because it was able to converge without problems. I found with my implementation that the fear that the designers of the Y-Means algorithm had that without a merging process, too many clusters would exist was unwarranted. Throughout testing, clearly defined clusters were kept as a single unit. It could be argued that in less clear portions of the example space over-splitting did occur, but this is of no real consequence because the only effect it would have is to lower the chosen value of p, the proportion of clusters marked as "normal," when the intrusion detection portion of the system takes effect.

2.2 Portnoy's Clustering Algorithm

The algorithm proposed by Leonid Portnoy [7] was the final clustering method to be implemented. A flow chart for the algorithm is shown in figure 3, to the right.

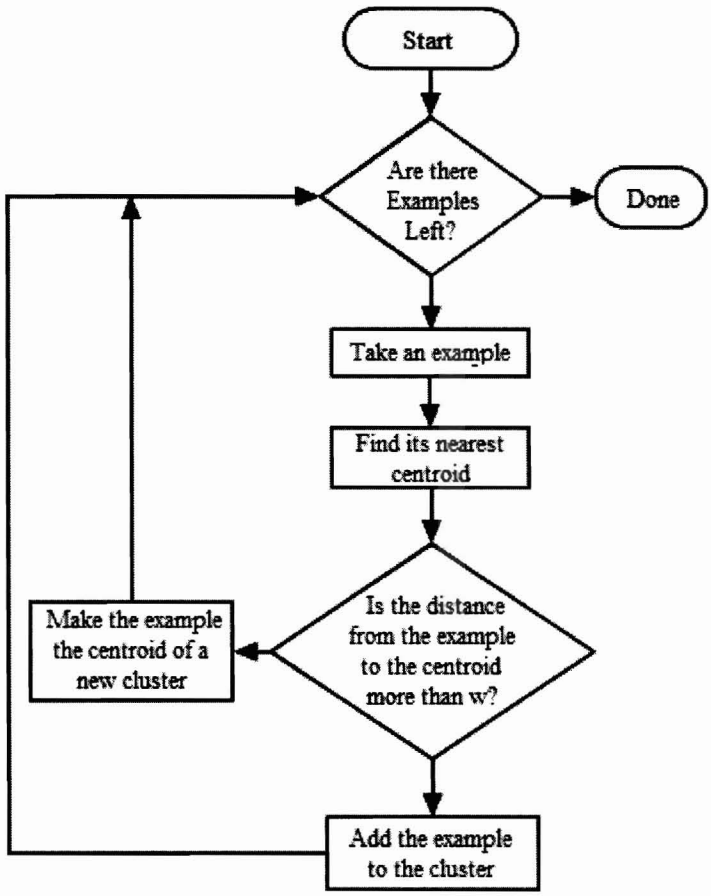

Figure 3: Portnoy's Clustering Algorithm

Portnoy's algorithm has a definite running time advantage over the K-Means and its variations. This is because the algorithm only requires one reading of the list of data examples. It works by taking each example one at a time, and placing it in the cluster of the centroid that is nearest to it. If, however, the example does not fall within the threshold of any cluster, then the example is used as the centroid of its own, new cluster.

The downside to Portnoy's clustering algorithm is that it is not as accurate as any of the previously discussed algorithms. This is because it is dependent on the order of the data examples. Obviously, a point that is in the middle of a cluster should serve as the centroid. But with Portnoy's algorithm, there is no guarantee of this.

2.3 The Standardized Space

For all of the clustering algorithms discussed, with the exception of K-Means, a constant cluster width value w is required in order for the algorithm to know when to create new clusters. This quickly brings up a few questions about the value. One must first, ask how w is determined. To this question, the general response is that different values of w must be tested until one is found that provides desirable detection and false-positive rates. The second major question pertains to fact that the choice for w seems directly related to the data. For instance, if an algorithm is asked to cluster four one-dimensional points: a=0, b=1, c=9, and d=10. It would seem obvious that these points should be grouped into two clusters and that an appropriate width to achieve this might be w=2. Thus points a and b would make up a cluster, as would points c and d. If the four points were instead a=0, b=10, c=90, and d=100, it would still seem obvious that they should be grouped the same way. However, using a value of w=2 for the width would result in four separate clusters.

In his paper, Leonid Portnoy proposes a technique, which he calls Normalization, for solving this problem. His idea is to use the standard deviation of each individual feature to perform a linear transformation that maps each example to a new standardized space. In order to do this, the average and standard deviation of the data must be generated. The average is the point in the same space as the original data, represented by a feature vector, where each feature is

the average of all of the feature values for all of the data. Mathematically, the average is represented as:

$$avg[j] = \frac{1}{n}\sum_{i=1}^{n} example_i[j]$$

where n is the total number of examples in the data set and $vector[j]$ means the jth feature in the feature vector.

The standard deviation, also represented as a feature vector, is generated from the average. Each element of the vector contains the standard deviation of that particular feature across the entire data set. The formula for generating standard deviation is:

$$stdDev[j] = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(example_i[j] - avg[j]\right)^2}$$

It is important to note Portnoy's choice to represent standard deviation as a vector. Some other implementations used standard deviation only as a single value. In doing so, however, valuable information was lost. It is important to store the standard deviation for each feature separately because this allows us to transform each point coordinate by coordinate to varying degrees. To demonstrate, let us look at the two-dimensional example of finding the standard deviation of the points (1, 1) and (3, 1). The average will be the point (2, 1) and the standard deviation will be (1, 0) since:

$$stdDev[0] = \sqrt{\frac{1}{2}\sum_{i=1}^{n}\left(example_i[0] - avg[0]\right)^2} = \sqrt{\frac{1}{2}\left((1-2)^2 + (3-2)^2\right)} = \sqrt{\frac{1}{2}\left(1^2 + 1^2\right)} = \sqrt{1} = 1$$

$$stdDev[1] = \sqrt{\frac{1}{2}\sum_{i=1}^{n}\left(example_i[1] - avg[1]\right)^2} = \sqrt{\frac{1}{2}\left((1-1)^2 + (1-1)^2\right)} = \sqrt{\frac{1}{2}\left(0^2 + 0^2\right)} = \sqrt{0} = 0$$

Once the standard deviation vector has been calculated, we are able to transform data points into the standardized space. The formula for carrying out this transformation is:

$$normalized\_point[j] = \frac{example[j] - avg[j]}{stdDev[j]}$$

This formula means that the transformation taking place is to graph each data point not by its original values, but by the number of standard deviations it is away from the average. So to continue with our example, the point (1, 1) will be transformed to $\left(\frac{1-2}{1}, \frac{1-1}{0}\right) = (-1,0)$ in the standardized space and the point (3, 1) will become $\left(\frac{3-2}{1}, \frac{1-1}{0}\right) = (1,0)$. Note that since division by zero cannot occur, any value of zero in the actual implementation of the standard deviation vector should be changed to a value that is extremely close to zero, such as 1×10^{-30}.

So, to see the effect, let us assume that a cluster width of w=0.5 is used to cluster the points given in the example. Since the points are more than half of one unit apart, they will be grouped separately, as we would hope. Now, let us assume a different set of points {(4.0, 0), (4.2, 0)}. If we use the same cluster width of w=0.5 to group these points, we will end up with both points in just one cluster since they are more than 0.5 units apart. This, however, is not what we want because, intuitively, if there are only two points, they should make up two groups. Now, if we use Portnoy's normalization process on the new set of points we will again get standardized points of (-1, 0) and (1, 0). Therefore, we can clearly see that using a cluster width of w=0.5 on both sets of points will achieve the desired result of two clusters for each.

2.3.1 Additional Effects of Normalization

The normalization technique proposed by Portnoy also has other effects not directly addressed in his paper. First, by performing this normalization, we are able to even out the effects of individual features on the measure of distance between points. Say, for example, that we have the data set {(0.01, 1.0), (0.02, 200.0)}. Without normalization, the second feature would completely dominate the measure of distance between these points. However, with normalization, these points will be translated to (-1, -1) and (1, 1) respectively, thereby weighting each feature appropriately so that the features have an even effect on distance.

The process of normalization also has the effect of exacerbating anomalies, which is an effect that is extremely helpful to us. To demonstrate, let us look at an example dataset of one-dimensional points that has 25 instances of the point a=(1.0), 24 instances of the point b=(1.001), and one instance of the point c=(1.2). The point a', which is the translation, or normalized version, of a, will be roughly (-0.160367631). Similarly, b'=(-0.124571284) and c'=(6.99890159). So, we can see that the distance between points a and b was originally 0.001, but after normalization, the distance is 0.035796347, which is a small change since the points started out very close. However, the distance between b and c, which started out as 0.199, has made a major change, to 6.87433031.

This technique was also important in coming up with a solution to the question of distance measures for discrete features addressed in the next section.

2.4 Measuring Distance

In these algorithms, there is much discussion about distance between examples in space. The way that distance was measured between examples for the purpose of clustering was to use

the Euclidean distance. The Euclidean distance between two points (p_1, p_2, …, p_n) and (q_1, q_2, …, q_n) is defined as:

$$\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

So, according to this, the distance between the points (1, 3, 2, 4) and (3, 4, 2, 1) would be:

$$\sqrt{(1-3)^2 + (3-4)^2 + (2-2)^2 + (4-1)^2} = \sqrt{(-2)^2 + 1^2 + 0^2 + 3^2} = \sqrt{14} \approx 3.741$$

This method is straightforward when all of the features in the example space are measured in continuous terms. However, the inevitable question arises as to how to deal with features that are measured in discrete terms. The method that was devised for this project's implementation was to simply define the subtraction operation for discrete terms as follows:

$$\text{For discrete terms a and b, } a - b = \begin{cases} 0, \textit{ if } a = b, \\ 1, \textit{ if } a \neq b. \end{cases}$$

A difference of 0 for equal terms is intuitively obvious because there should be no distance between things that are the same. However, the arbitrary selection of a difference of 1 for unequal discrete terms may not seem to make sense. While it is true that the value of 1 is arbitrary, it should not matter when we are measuring distance. This is because of the transformation process taken from the paper by Portnoy. Because this we know that this transformation will occur, it does not matter what the value of the difference is, so long is it is not 0, because it will be scaled according to the standard deviation of the entire data set so that it has the appropriate effect on the distance measure.

We can see how this modified version of Euclidean distance works but looking at an example of the difference between two points with discrete terms. According to this modified definition, the distance between the points (1, true, 2, true) and (3, true, 2, false) would be:

$$\sqrt{(1-3)^2 + (true - true)^2 + (2-2)^2 + (true - false)^2} = \sqrt{(-2)^2 + 0^2 + 0^2 + 1^2} = \sqrt{5} \approx 2.236$$

3 Clustering for Intrusion Detection

The clustering of data does not automatically lend itself to classification for intrusion detection. The purpose of intrusion detection is to label a particular example as either normal or anomalous. Clustering alone, however, does not perform a binary classification. Therefore, we must convert the many clusters generated by the algorithm into two distinct groups. We are able to do this with a few important assumptions. The first assumption is that similar traffic will be spatially located close together. This means that when we apply a clustering algorithm to our data, similar traffic will most likely be clustered together. This will result in relatively homogenous clusters. This is important because in order for clustering to help us detect intrusions, we must know that, for the most part, normal traffic will be clustered with other normal traffic and intrusive traffic will be clustered with other intrusive traffic.

The second assumption is that the amount of normal traffic is significantly higher than the amount of intrusive traffic. This assumption is important to us because it means that clusters of intrusion examples will be very small when compared to clusters of normal examples. Using the process described below, this will allow us to decide which clusters are of normal traffic and which are of intrusive traffic without knowing the actual labels for the examples.

Using these assumptions we can formulate a strategy for using clustering as a method to detect intrusions. We do this by choosing a constant p to represent the percentage of clusters that are normal. After applying a clustering algorithm to the data to create many clusters, we order the clusters by the number of examples that each contains. We can then choose the largest p percent of clusters to label as "normal" and label the remaining clusters as "anomalous." This will give us the template which we can now use to label new examples.

In order to label a new example, we must start by graphing that example in our example space. We should then find the cluster out of all the clusters in the template whose centroid is closest to the example. We can then label the example with the same label as the cluster. Therefore, in essence, any new example takes on the label of the cluster it is nearest to. This makes sense in context because we would assume that any new example would be similar to those examples that are spatially near to it, and thus should have the same label as those nearby examples.

This process led to a modification that does not seem to be specifically addressed in other papers. While it certainly makes sense to mark a new example with the same label as its nearest cluster, it seems that there should be some contingency for those new examples that are outside the width of any cluster. The reason for this is that it might be possible for a new example to be very far from any cluster, but *closest* to a cluster labeled "normal." With the previously described algorithm implemented strictly, that example would be marked as "normal" even though, instinctively, it should be assumed that an example that is far from anything should be marked as an intrusion because, by definition, it is anomalous. Obviously, if a new example is far from any cluster but nearest to a cluster marked "anomaly," then there will not be a problem because the new example will be rightly labeled as an intrusion.

The modification that was added to the implementation was a check when the nearest cluster centroid is found. By using the same cluster width constant w as previously described, we can add the condition that if the distance from the new example to its nearest cluster centroid is greater than w, then that new example should be marked as an "intrusion," regardless of the cluster's label. However, in running tests, the improvement made by adding this feature was negligible. This is presumably due to at least one of two situations. First, it is possible that the

template is encompassing enough that it is very rare for any new example to be outside of *every* cluster. This would result in nearly every new example being within the threshold of a cluster, meaning that there would be no difference in labeling between the two approaches. The other reason that there was no significant improvement could be that the cluster or clusters labeled "normal" could be fully or near-fully surrounded by clusters labeled "anomaly." This would mean that even if a new example were outside the threshold of any cluster, it would be highly unlikely or impossible for that example to be closer to a cluster labeled "normal" than a cluster labeled "anomalous."

3.1 Evaluating Detection Accuracy

In order to measure the accuracy of a clustering algorithm, a pre-labeled dataset is required. The dataset that was used for testing was the KDD Cup 1999 data set [7]. The testing process was very straightforward. First, a template was generated using one or more of the techniques previously outlined. Then each example was read from the test dataset into my program and graphed in the template's example space. Then the algorithm was able to label each test example as either "normal" or "intrusive" based on its position in relation to the clusters in the template. Finally, the label generated by the clustering algorithm for the example could be compared to the label provided with the example. However, tests resulting in responses of either "right" or "wrong" are not descriptive enough to provide the type of analysis we want to measure the accuracy of our system. We need to be sure that our measure of accuracy will indicate the effectiveness of a particular algorithm in a real world intrusion detection setting. Therefore, we break the results of a test down into four categories:

1. True Positive: An example of intrusive traffic, marked as "intrusion" by the algorithm.
2. True Negative: An example of normal traffic, marked as "normal" by the algorithm.
3. False Positive: An example of normal traffic, marked as "intrusion" by the algorithm.
4. False Negative: An example of intrusive traffic, marked as "normal" by the algorithm. Or in other words, this is an intrusion that was not detected by the system.

We can then use these types of results to construct a two-part measure of accuracy that will help us to analyze the effectiveness of the clustering algorithm.

There are two numbers that must be considered when analyzing the accuracy of an intrusion detection system. The first is the detection rate, which is defined as follows:

$$DetectionRate = \frac{TruePositives}{(FalseNegatives + TruePositives)}$$

This equation is equivalent to saying that the Detection Rate is the proportion of True Positives to the total number of intrusive examples. Put more plainly, it is the percentage of intrusions that are detected out of all of the intrusions that pass through the system. This measure tells us whether we are actually detecting an appropriate number of intrusions with our system.

The second number used in measurement is the false positive rate, defined as follows:

$$FalsePositiveRate = \frac{FalsePositives}{(FalsePositives + TrueNegatives)}$$

The false positive rate is the proportion of test examples falsely marked as "intrusive" to the total amount of normal traffic. Or, in other words, it is the percentage of the instances of normal traffic that are thought by the system to be intrusive traffic.

Both of these measures are necessary in determining whether an algorithm is suited to use in an intrusion detection system. The detection rate indicates how much of the intrusive traffic is

detected as intrusive. This is obviously important because it is worthless to have an intrusion detection system that fails to detect a majority of intrusions. It is less obvious why the false positive rate is important because it seems at first look that we would only be concerned with whether or not the system is detecting intrusions. However, if the system marks too many instances of normal traffic as "intrusive," then network security administrators will spend a great deal of time wading through false reports of intrusions, which would probably result in less attention paid to reports of intrusion when they arise in the future.

3.2 Passing Information to Enhance Detection Accuracy

It was the goal, through the course of this research, to enhance current clustering techniques in order to improve accuracy. The method that was explored along this route was that of how distance is measured in the clustering algorithms. Euclidean distance is the most popular choice for a distance measure, but it is certainly not the only choice. An entirely different distance algorithm was not used, but instead the possibility for skewing the Euclidean distance between points was examined. It was hoped that previously learned knowledge could be used to enhance the distance measures to improve the overall accuracy of the intrusion detection system.

It was desired that the advantages of using entirely unlabeled data that were discussed earlier be maintained. The method that was developed to pass data while still avoiding using expertly generated labels was to run the intrusion detection system successive times, but remember the results of each previous run. The way that was devised to have previous knowledge impact the next run was to stretch or shrink the distances between point based on whether they were marked as anomalous or normal. Therefore, a modified formula for distance was created:

$$\sqrt{\sum_{i=1}^{n}\left((point_A[i]-point_B[i])\bullet scale\_factor\bullet\left(\frac{1}{scale\_factor}\right)^{2-2*P(point_B[i])}\right)^2}$$

In this formula, the value of P is the proportion of examples with the particular feature value $point_B[i]$ that were marked as anomalous in the previous iteration's result. The scale factor is some constant value that decides how much stretching or shrinking is to occur. The additional portion of the formula works in this way: if the proportion P for the feature value $point_B[i]$ is less than 50 percent, then it means that less than 50 percent of the examples with that feature value were marked as anomalies. For this we could infer that the example we are currently looking at, because it has feature value $point_B[i]$, is probably not an anomaly. The smaller the percentage for P($point_B[i]$), the more likely it is that the current example is not an anomaly. For features where P($point_B[i]$) is less than 50 percent, the difference between $point_A[i]$ and $point_B[i]$ is shrunken. If P($point_B[i]$) is zero, meaning that none of the examples with that particular feature value were marked as anomalous, then the difference will be shrunken to the most extreme degree allowed, which is to be divided by the scale factor. The same line of logic can be used for values of P that are greater than 50 percent. In these cases, the difference between $point_A[i]$ and $point_B[i]$ is stretched. In the case where P($point_B[i]$) is one, meaning that all of the examples with that particular feature value were marked as anomalous, then the difference will be stretched to the most extreme degree allowed, which is to be multiplied by the scale factor. The graph to the right shows the degree to which the difference between feature values is altered for any

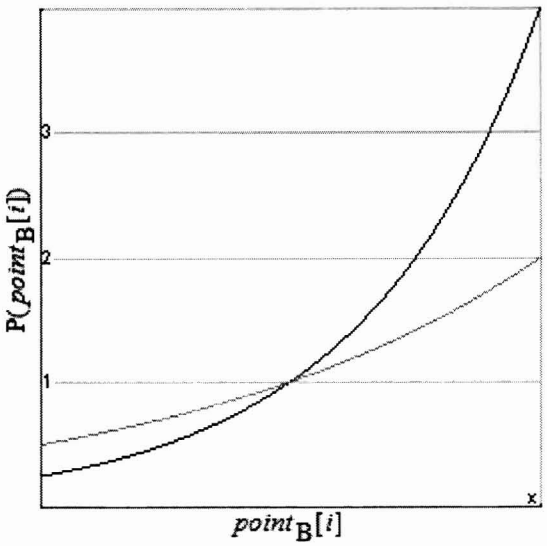

Figure 4: Amount of distance skew

value of P. The x-coordinate of the graph represents P, going from zero to one. The y-coordinate represents the scale factor. The lines show the value that the difference between feature values is multiplied by to achieve the altered distance. The steeper line is the graph when the scale factor is four. The flatter line is of when the scale factor is two. As the graph shows, the higher the likelihood that a particular feature value appears in an example, the more that feature will increase the overall distance measure between points. And for any scale factor, if the value of P is 50 percent, then the difference is left unchanged.

Determining the proportion of a discrete feature value that appears in examples marked "intrusion" is fairly straightforward. The program is able to easily count the number of times the feature value appears and how many of those times are in examples marked "intrusion." Continuous features, on the other hand, were more complicated. The method devised to evaluate the correlation between a particular continuous feature value and its being in an anomalous example was to break the range of continuous data into N sections and to find the average proportion P for all of the feature values that appear in the same section as the continuous feature being analyzed. This is shown in the graph in figure 5, to the right, where the top of the rectangle is the value used as the proportion of times a feature value that falls in that rectangle occurs in an anomaly. It is important to note that the more sections the continuous feature value space is divided into, the more accurate the algorithm will be. Notice that in the graph, the

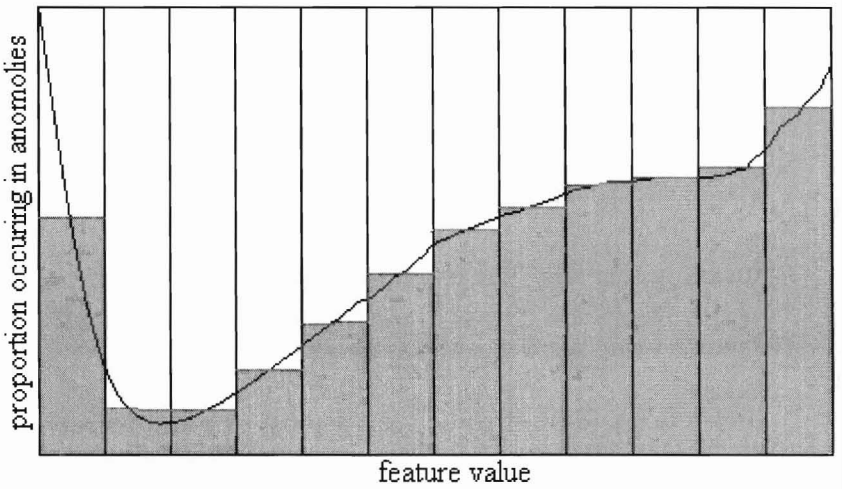

Figure 5: Graph of continuous feature value anomaly proportions

first rectangle is the average of all the proportions in it, but it is not a very accurate measure for those feature values at the extremes of the section's width. However, when the number of sections increases, so does the amount of time it takes for the program to look up the value. Therefore, when implementing this algorithm, care should be taken to choose a number of sections large enough to give good results, but not so large that the program is slowed too much.

The logic behind this approach derives from the assumption that anomalous data is "far away" and that normal data is "close." When the Portnoy algorithm is deciding whether a particular data example should be inserted into a given cluster, it measures the distance between them and checks whether that distance is within the threshold. If the cluster is of normal data, then would want the distance measured between it and a normal example to be small so that the normal example is inserted into the cluster. If the example is from an intrusion, we would want the distance measured to be large, so that the example would not be placed in the cluster with the normal data. By skewing the distances with known information, it was hoped that this would be able to increase the likelihood that normal data is placed in normal clusters and abnormal data is not put in them.

While it is true that one could generate the proportions for each feature labeled anomalous or normal from the labels provided with the data set, we would like to be able to avoid this because, again, this is a use of expert knowledge that we do not want to rely upon in a real world IDS situation. Therefore, the technique devised to accomplish this without labeled data was to run the IDS algorithm iteratively. After each iteration, the program calculated the proportions for each feature labeled anomalous and normal so that this information could be used to adjust the distance measures in the next iteration.

4 Test Results

4.1 Results from Tests on the Portnoy Algorithm

The results from the test runs of the implementation of Portnoy's clustering algorithm yielded good results. The first set of tests was used to determine what the best cluster width would be. Several test runs were performed with a constant "normal proportion" of 1.0% and varying cluster widths. The results, shown in table 1, indicated that a cluster width of about 10.0 was optimum.

Cluster Width	5	10	15	20
DR (FPR)	79.5 (7.014)	78.4 (3.29)	25.3 (0.815)	7.56 (0.47)

Table 1: Tests to Determine Cluster Width

The next set of test was used to determine the optimum "normal proportion." I used a constant cluster width of 10.0 and varied the proportion. The results, as shown in table 2, indicated that a "normal proportion" of about 1.5% was optimal.

Proportion	1.0%	1.5%	2.0%	15.0%
DR (FPR)	78.4 (3.29)	78.0 (2.46)	25.3 (1.27)	1.48 (0.19)

Table 2: Tests to Determine "Normal Proportion"

I also performed tests to see if marking all outliers as "intrusion" would result in better performance. The results, in table 3, show a small increase in detection accuracy.

Normal	78.0 (2.46)
With marking outliers as "intrusion"	79.2 (2.45)

Table 3: Tests of Marking All Outliers as "Intrusion"

4.2 Results from Information Passing Tests

The results from the test runs of my implementation of this distance skewing mechanism showed not an increase in accuracy with each iteration, but instead, a dramatic decrease. More investigation will be required to determine precisely why this was the case. One possibility could be that, in my implementation, when distance was being measured from a cluster to a new

example, only the proportion P from the example was used to skew the distance. In other words, the likelihood of the cluster's center to be an anomaly was not considered. It might, however, be more effective to use the proportions from both the cluster's center and the example being inserted. In order to do this, the distance equation would have to be altered to include the value of $P(point_A[i])$.

5 Possible Areas for Further Exploration

5.1 Automatic generation of "normal proportion"

Any intrusion detection system will be most useful with as little expert knowledge required as possible. We have addressed this issue by choosing to use unsupervised learning techniques so that the system is more autonomous. However, we are still required to supply values for the cluster width, w, and the proportion of clusters that should be marked as "normal", p. The value chosen for the cluster width w seems to have less importance if the example space transformation technique introduced by Portnoy is used because it seems that w can be a fixed value that will not need to be changed by the end user. However, it is not clear that the "normal proportion" p can be fixed in the same way. It might be worth exploring, therefore, whether there is a way to choose which clusters are "normal." The way that I thought of, but have not yet tried is to separate the largest clusters from the majority of clusters based on standard deviation of cluster size. For instance, if a cluster is a size of at least x standard deviations, then that cluster should be marked "normal."

5.2 Fixing the Portnoy algorithm's dependence on data order

Another possible area for further exploration is searching for ways to reduce or eliminate the inherent flaws in Portnoy's clustering algorithm. The accuracy of clustering will always be undermined in this algorithm because of the dependence on the ordering of the data. If there were a way to eliminate this dependence, while still maintaining the major advantage in time complexity over K-Means, then this could greatly improve the clustering process. The simple method that I came up with to accomplish this was, after each new example is inserted into an existing cluster, to move that cluster's centroid to its new average. Doing this should, theoretically, eliminate the possibility that a cluster's centroid might be far away from the true average of the cluster, and thus skewing the way that the algorithm clusters the data. Also, it will only increase the algorithm's time complexity slightly. While Portnoy's original algorithm only ran through the list of examples once, this change will only add, at most, the run-through of one cluster per example added. In an example created for demonstration, the proposed modification did indeed solve the problem. The K-Means algorithm still had a slightly different result, but certainly not major enough to warrant the huge increase in running time.

6 Conclusions

Intrusion detection will continue to be an important area in the field of information technology because it is clear that the problem of computer hackers is one that is not going to disappear. It also seems evident that anomaly detection will become an increasingly important part of the intrusion detection field for the reasons outlined above. The research done for this project has confirmed the usefulness of clustering as a tool for use in an intrusion detection system. Though the distance skewing method proposed did not yield good results, the general

concept should not be ruled out entirely. Also, the questions posed in this paper are worthy of further exploration and could potentially increase the effectiveness of clustering even more.

7 Acknowledgements

I would like to thank Dr. Jinqiao Yu for his assistance as my advisor for this research project, Dr. Weiyu Zhu for his expertise in the area of Machine Learning, and Dr. Hans-Jörg Tiede for his guidance as my academic advisor. In addition, I would to thank these three professors from the Illinois Wesleyan University computer science department, along with Dr. Michael Dancs of the Illinois Wesleyan University mathematics department for participating in the defense committee for this thesis.

8 References

[1] Alpaydin, Ethem. Introduction to Machine Learning (Adaptive Computation and Machine Learning). Boston: The MIT Press, 2004. 1-20.

[2] Axelsson, Stefan. "Intrusion Detection Systems: a Survey and Taxonomy." 14 Mar. 2000. Department of Computer Engineering, Chalmers University of Technology. 12 Apr. 2006 <http://www.mnlab.cs.depaul.edu/seminar/spr2003/IDSSurvey.pdf>.

[3] Frank, Jeremy. "Machine Learning and Intrusion Detection: Current and Future Directions." 9 June 1994. Division of Computer Science, University of California at Davis. 12 Apr. 2006 <http://ic.arc.nasa.gov/people/frank/ncsc.94.ps>.

[4] Guan, Yu, Ali A. Ghorbani, and Nabil Belacel. Y-Means: a Clustering Method for Intrusion Detection. Canadian Conference on Electrical and Computer

Engineering, 3 May 2003, National Research Council Canada. 12 Apr. 2006 <http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-45842.pdf>.

[5] Jones, Anita K., and Robert S. Sielken. "Computer System Intrusion Detection: a Survey." 9 Feb. 2000. Department of Computer Science, University of Virginia. 12 Apr. 2006 <http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-survey-v11.pdf>.

[6] "KDD Cup 1999 Data." UCI KDD Archive. 28 Oct. 1999. Donald Bren School of Information and Computer Sciences, University of California, Irvine. 12 Apr. 2006 <http://www.ics.uci.edu/~kdd/databases/kddcup99/kddcup99.html>.

[7] Portnoy, Leonid. Intrusion Detection with Unlabeled Data Using Clustering. Diss. Columbia Univ., 2001. 12 Apr. 2006 <http://www1.cs.columbia.edu/ids/publications/cluster-thesis00.pdf>