Data Mining Report

Department of Computer Science and Engineering
College of Engineering
University of Nevada, Reno
Juliet Villanueva (50%), Eric Garza (50%)
Dr. Lei Yang
December 12, 2017

1 Introduction

In today's society, data is everything. However, with the amount of data that is generated, it is difficult to understand and actually make use of the data that is available. This is when data mining becomes applicable. Data mining takes large amounts of data and creates some kind of prediction for the future or details different patterns. In simple terms, data mining takes large unusable data and makes that data useful. Using algorithms for data mining could potentially save a life, by finding abnormal heart problems when hooked up to a heart monitor. It could increase revenue for a company by specify items to put on sale based on the season, time of year, holidays, or customer's that they normally receive. It could potentially give information on whether there is a fire in the area or not. The amazing thing with data mining is that it gives meaning to large amounts of data and provides information that we can use as a society to help with the quality of life. This project detailed a few of the data mining techniques applied to a data set that was extremely sparse. Each technique had varying accuracy with different types of data. In the tests that were run, the accuracy of the techniques were calculated and compared.

2 Literature Review

The classification techniques that were used in our project were K Nearest Neighbors, Neural Networks, and Support Vector Machines. Each of these classifications take different approaches when decoding the data. The description of each technique is explained below.

2.1 K Nearest Neighbors

K Nearest Neighbors groups the data into clusters of similar data. This is achieved through a distance metric and the number of clusters that are going to be created which is represented by a number K. The distance from one data point to another is calculated from the euclidean distance formula given below.

$$d(p,q) = \sqrt{\sum_{i=1}(p_i - q_i)^2} \tag{1}$$

All of the data points that are within a certain distance to the cluster become part of that cluster. In order to determine which class the cluster will be, the cluster takes a majority vote. However, the data points that are closer to the center of the cluster have a larger weight for the vote then the data points that are on the outer edges of the cluster.

In order to achieve an accurate classification when using K Nearest Neighbors, one would have to know the number of clusters that they want the dataset to group the data in. However, with an unknown dataset it is very difficult to decide the number of clusters that the data should split into. Although, the question comes to mind, why is this important? If K is chosen to be the same number of classes, shouldnâĂŹt that work? However, choosing a K that matches the number of classes is not something that will work with every dataset. It might work if each class is densely concentrated in one place away from another class, which is the case in Figure 1.

However, with most datasets this is not the case, one class could be located in many different clusters and could be spread across the graph, similar to Figure 2. Choosing a small K makes the clusters prone to noise and choosing a K that is too large could make the error large. Therefore, one disadvantage to using K Nearest Neighbors is choosing a correct K value.

Another disadvantage is that K Nearest Neighbors is considered a lazy learner. Which means it does not

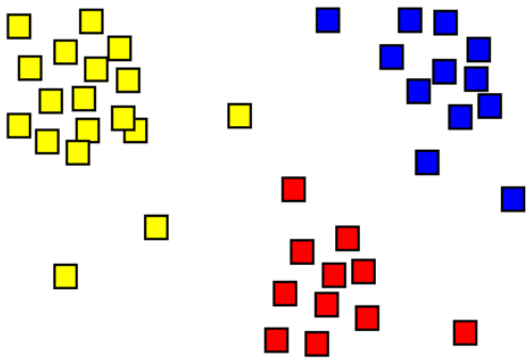

Figure 1: *An example of K Nearest Neighbors. K in this situation equals three, which also equals the number of classes. In this situation the data points are concentrated near similar classes. This would be an ideal dataset, however this is unrealistic data.*

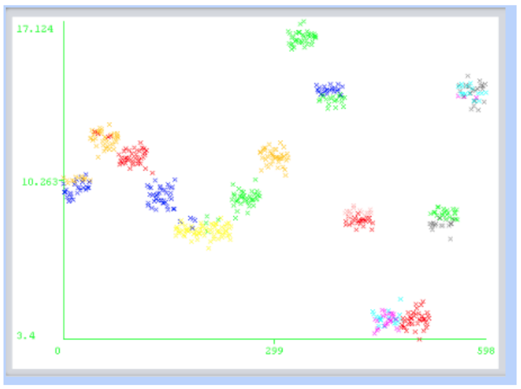

Figure 2: *More realistic looking data. There are clusters all over the graph, however, there are only two classes for this data.*

create a model for the data and therefore makes classifying unknown records very expensive. An example of an eager learner would be Artificial Neural Networks, which will be discussed in the next portion.

2.2 Artificial Neural Network

Many People try to compare Artificial Neural Networks (ANN) to actual neural networks in the human brain and in a way they are similar. ANNs are a connected network that take input and give an output, as shown in Figure 3. However, it does not just fire off information to the next node or neuron, each neuron has many different elements.

Each input into the neuron has the actual input as well as a weighted edge associated with that input. The neuron takes all of the inputs and sums them together, then it passes that value to an equation, called an activation function. The output is what is given from the activation function with the weighted input values. Once the output is calculated, then it could either go to other neurons, or it could give the actual value. The entire process for a neuron can be seen in Figure 4.

For the case of this project, Multi-layered Perceptrons(MLP) were used. Multi-layers are described above as including hidden layers, so what is a perceptron? Perceptrons are feed-forward networks, which basically means each output is the input for another layer; there are no cycles between layers of neurons. Since

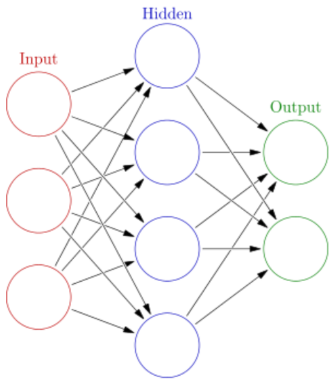

Figure 3: *An example of an Artificial Neural network. There are input layers, hidden layers, and finally the output layer.*

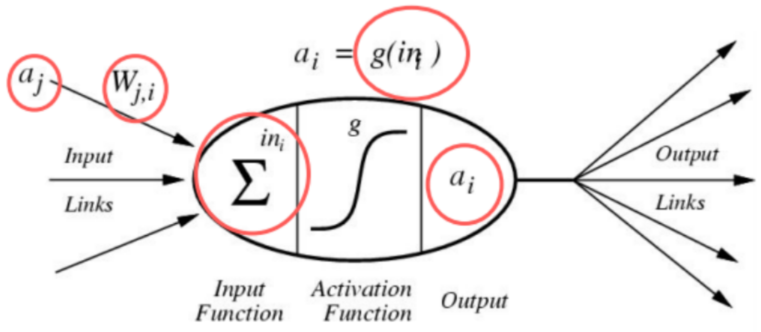

Figure 4: *A neuron within an Artificial Neural Network. This neuron takes inputs, along with their weighted edge values and sums them together. Then uses the summation as a variable in the activation function, which yields the output.*

the algorithm is multi-layered, it makes computing difficult problems, much easier. In the case that the data could not just be separated linearly, the MLPs are still able to create a distinction between the two different classes. Another algorithm that is able to make distinctions between classes, even if it is not linearly separable, is a support vector machine.

2.3 Support Vector Machine

A support Vector Machine (SVM) creates a hyperplane to separate the data. The algorithm chooses the hyperplane that creates the largest margin between the data. A margin is the space between the two sets of data and is created using two data points from each class, known as support vectors. These two data points are on the boundaries of their classes and close to the other class. Using these two support vectors, the margin can be created by testing different hyperplanes and finding the largest distance between the two support vectors. This can be done linearly or nonlinearly. Nonlinear classification is when the classes are split using nonlinear kernels, such as a Gaussian split. The advantage to using this kind of split is the classes could be mixed together, but the split would still have a pretty accurate reading. An example of nonlinear SVM is shown in Figure 5.

A linear classification will create a straight line between the sets of data, this line will have a maximum distance between the sets of data, an example is shown in figure 6. However, in this case, the data has to be

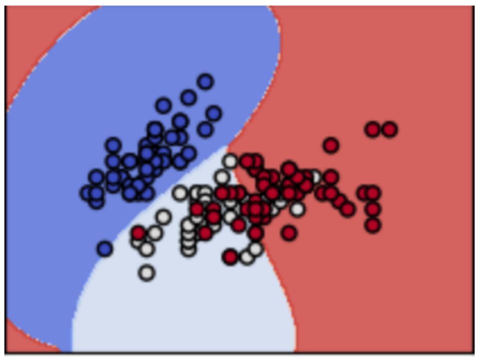

Figure 5: *An example of nonlinear SMV. The splits curve to get the largest margin between the data sets.*

separable linearly. Meaning there has to be a clear boundary between the sets of data, otherwise it would be difficult to use this classification.

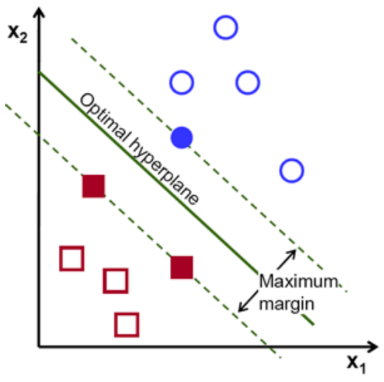

Figure 6: *An example of linear SVM. The filled in shapes are the support vectors. Using those data points, hyperplanes were created that maximizes the margin value.*

However, linear SVM does work very well with sparse data, since it is only splitting between classes, using one data point, the support vector, and not considering all the data points as a whole. With this in mind, and the fact that the project had a very sparse dataset, linearSVC was one of the first techniques chosen.

3 Task 1

In task 1, we were provided with three documents. The first document was the training data, which held over 1,800 different dataset ID's and over 25,000 features ID's. The data was given in three columns: the dataset ID, the feature ID, and the feature value. The reason for the strange format was because the data was incredibly sparse. In most cases, the features were not included in over half of the dataset ID's. The second document was the classification's for the training data. These were given as values of either 1 or -1, which meant that there were only two classes. Finally, in the third document, the testing set was included. The testing set was given in the same format as the training data. With these three documents, the task was to use the best classification algorithm to get the best classification results for the testing data. Along with classification, preprocessing needed to be done as well. The details of the approach that was taken is discussed further below.

3.1 Approach

With the data given, our group decided to use python to implement the classification. Python has several libraries that work extremely well with data mining classifications. One of those libraries is numpy. Numpy is a math library, which was used to store the data in a matrix. The matrix data structure from Numpy allows easy use with the SciKit Learn library. SciKit Learn has many different classifiers as well as preprocessing techniques. It is the perfect library to use for data mining classification.

3.1.1 Preprocessing

The preprocessing technique chosen for feature selection was Variance Threshold. Variance is how similar two data points are to one another. If the variance is low, then that means that the two data points are extremely similar. Thus all the data with variance that was below our threshold was removed. The threshold that we chose was 0.05. This number was chosen after many different trial runs. In the first few trial runs, we set the threshold to 0.5. However, that removed more than 99% on the features. When using this threshold, the classification accuracies that we got were low. Lowering the threshold meant that we would keep more of the features, meaning more training data for the classification algorithms. Each algorithm needs a certain amount of data in order to actually create a model to run tests on. Although one of the classification algorithms didn't change with the threshold. K Nearest Neighbors remained the same when the threshold was decreased, however the other two classification accuracies increased. ANNs increased the most with a 30% increase between the 0.5 threshold and the 0.05 threshold.

3.1.2 Model Selection

Selecting the model to use for training was done on a 80/20 split. The algorithm splits the training data into 80 percent training and 20 percent testing. This model was used in order to get a proper accuracy on the algorithms that were run. With 20 percent of the training data going toward testing, we were able to use the labels to check if the data was accurately labeled in the training set.

3.1.3 Parameter Selection

Each of the classification algorithms used had a large number of parameters that could be used or changed. They all also had default values for the parameters, so they could be run without having to change the parameters. When different parameters were changed in each of the algorithms the accuracy actually went down. Therefore for this project, the default parameters were used to create classifiers.

3.1.4 Solution

For this project, there were three algorithms tested. The three algorithms consisted of K Nearest Neighbors, Artificial Neural Networks, and Support Vector Machines. Each of the algorithms had very different outcomes when they ran the training data. After the training data was split into 80% training data and 20% testing data, models were made using each of the algorithms. Once the models were made the last 20 percent of the training data was used to test the accuracy of the model. K Nearest Neighbor had fairly good accuracy, it was approximately 65% with a standard deviation of +/- 0.01. The next best was the Artificial Neural Network with approximately 93% and a standard deviation of +/- 0.08. Finally the algorithm that had the best accuracy was the Support Vector Machine, and in this particular case the linear SVM. It ran with an accuracy of approximately 95% and a standard deviation of +/- 0.05. Since the linear SVM ran with the highest accuracy we decided to run the testing data with that algorithm.

4 Conclusion

This project saw a few data mining techniques in action and how well they performed against each other. Task 1 compared various different data mining algorithms including K Nearest Neighbor, Artificial Neural Network, and Support Vector machine. It was a solid introduction into the world of data mining and was a great way to implement techniques learned throughout the course. Future work of this project could test more algorithms to see how well they perform against each other or maybe even conduct an experiment that would include gathering of the data instead of having it provided.