Competitive Project (Machine Learning)

Name: Spencer Fronberg
UID: u0766439

December 15, 2017

1 Algorithms Worked on / Why interesting

For this competitive project, I worked on many different algorithms. I worked on Decision Tree, Bagged Forest, SVM, Logistic Regression, Aggressive Perceptron, and Perceptron with Dynamic Learning Rate. I tried to implement other algorithms also like SVM over trees and Logistic Regression over trees but for some reason I was not getting very good accuracy because each of my trees would predict the exact same label for each example which does not mean that all examples were predicted the same, just that all trees predicted the same for each example. I think that because it was predicting the exact same result for each example, when I created a vector for each example of size 1000 (size of forest), there was no learning when I ran SVM or Logistic Regression. When it came time for me to decide if I wanted to try Naive Bayes, I realized that it was not the best algorithm from homework #5, so these are my reasons to why I decided to chose the algorithms I chose.

I think that each of these algorithms that I chose are interesting in many different ways. The main reason why I think it is very interesting is you can take a very simple type of algorithm that makes very simple computations, and you can get very good accuracies with the data. On top of that, it is interesting because these algorithms can be used in real life problems. We could take these algorithms and determine if an account on twitter is spam (as in this project) or determine how to categorize articles. These algorithms can be used in many different topics.

2 Important Ideas Explored

Like I said before, I tried to explore SVM and LR over trees, but they did not work. For Decision Tree, I decided to take the best decision tree that was created from cross validation. When I did this I realized that this was better than cross validating and using the best depth to train a new tree. I received slightly better accuracy getting the best tree from cross validating with different depths. Also with decision tree, I tried something similar to bagged forest before I even did bagged forest. What I tried was taking the average voted prediction from the five separate trees that were created from the five fold cross validation. Doing this did not increase the accuracy. For Decision Tree, I also explored adding two additional features to the tree. The first feature was the total amount of re-tweets each user had done. The second was the total number of tweets each user has tagged at least one user in their tweet. When I made these additions, the first feature actually increased my accuracy, but the second feature made my accuracy worse. With that, I decided to keep the first feature and discard the second.

When it came to bagged forest, I still cross validated for the best depth. For the number of examples to train on each tree, I tried many different possibilities like 100, 150, 1000 1500, and 14500. I realized that training each tree on a random 1500 examples gave me the best accuracy. For both Logistic Regression and SVM, whenever I made an update to an element in the weight vector, I decided to only update that element in the weight vector if the x_i was not equal to 0. I tried this out to see if I would get better accuracy and surprisingly I did so I decided to keep it implemented. For both Aggressive Perceptron and Perceptron over Dynamic Learning Rate, I decided to try out and see if I could get better accuracy if I created 500 weight vectors and bias's based off of a different random seed for each weight vector and bias. I realized doing this did not help the algorithm much.

While cross validating for the best hyper parameters for each algorithm, I explored combining the training set and test set, shuffling the data multiple times, and dividing the data up into five separate data's for cross validation. Doing this actually gave me slightly better accuracy.

3 Ideas From Class

When it came to after we had already cross validated and we were storing the accuracy, the weight, and bias, I was initially confused how I should have done it. I was initially storing it in a dictionary where the key was the accuracy and the value was the weight and bias object, but I realized that sometimes the accuracies were the same in some of the epochs, so in class I was talking to one of the TA's and they said to not store the accuracy as the key in the dictionary, so I then realized that I could make the key for the dictionary the epoch number and the value would be an object that stored the Accuracy, Weight, and Bias. I took what I learned in class from the TA and made the Perceptron Algorithm (two of my algorithms) work. When it came to implementing Naive Bayes from homework #5, I took the idea from class of taking the log of each literal and summing them up. If I would have done this algorithm, I would have implemented it for this project also because it would have improved the accuracy for Naive Bayes. Using this same idea of taking the log, I tried taking the log of each of the feature values for each example in all datasets for SVM, Logistic Regression, Aggressive Perceptron, and Perceptron with Dynamic Learning Rate. Doing this only made my accuracy worse than it originally was, so I decided to not keep it implemented in the code.

With this project the main ideas that I used was how I cross validated. For each algorithm, I cross validated with either one or two variables which we learned in class to do.

I also took the right thresholds for each variable. For bagged forest and decision tree, my hyper parameter was depth and I set it at a range from 2 to 9. For SVM my two hyper parameters were Learning Rate and Tradeoff which were both from a range of 10 to 0.0001 (powers of ten). For Logistic Regression my Learning Rate range was the same as SVMs, but my Tradeoff was from 0.1 to 10000 (powers of ten). For both Perceptron over Dynamic Learning Rate and Aggressive Perceptron, my Learning Rate ranges were both form 0.01 to 1 (powers of ten). Because my Aggressives best Learning Rate was 0.1, I did not need to test for lower values because it was in the middle of the values I tested which the professor said in class that the accuracy will go up in the middle where the best hyper parameter is and curve down as it gets farther away from the best hyper parameter.

4 What I learned

There are many things that I would say that I learned from completing this project. The main thing that I learned is that overall using Decision Trees is one of the best type of algorithms that you can use over SVM, Logistic Regression, Perceptron, and most of the other algorithms when it comes to this dataset. Doing bagged forest over decision tree can improve the accuracy of Decision Tree and most likely it will increase it. I also learned that the best algorithm can vary when it comes to different types of data sets. So sometimes Logistic Regression could be better than Decision Tree, but most of the time Decision Tree will be one of the best algorithms from most of the datasets. I know that there are probably better algorithms than Decision Tree that I did not use, but out of all the algorithms that I have done Decision Tree is the best for this dataset (includes Bagged Forest). I do believe though that if I would have been able to get SVM and Logistic Regression over Trees working, that those would give better results but SVM over Trees, Logistic Regression over Trees, and Bagged Forest are all versions of Decision Tree. That is why I say that Decision Tree usually gives one of the best accuracies. Overall, Support Vector Machines (SVM), Logistic Regression (LR), and Perceptron are all good algorithms, but the best for this dataset is Decision Tree.

I also learned the adding a feature to your Decision Tree will either increase your accuracy, decrease it, or keep it the same, but I know for sure that if you add the correct feature to your Decision Tree, then your accuracy should go up, but that for sure depends on whether or not you have too many features already. Sometimes too many features in your dataset can decrease your accuracy. The amount of thresholds that you decide to choose for each feature is a major factor of whether or not your accuracy goes up or down. On top of that you need to make sure that your ranges for each threshold has the best range. Some features need the first few ranges at a very low number increasing the ranges as you get higher and some need the ranges to be equal. Cross Validation also plays a huge role when it comes to any algorithm. Without cross validation our accuracies would not reach as high as they reach.

When training and testing your data, you need to make sure that you do not train your data with both the Training and Test set because that can corrupt classifier. You also want to make sure that you are not overfitting your classifier. This happens when your classifier represents all the noise in the training data. It can also happen when your training data is too small. To be able to detect if this is happening, all you need to is ask yourself one question. Am I getting a good accuracy with the training set and not getting good accuracy with the test set? This is the main warning to detect overfitting.

5 Results

The table below shows my accuracies for each algorithm and the best hyper parameter(s). The percentage that is in parenthesis by the name of the algorithm is my Kaggle submission accuracy.

Algorithm (Kaggle Accuracy)	Best hyper-parameters	Average cross-validation accuracy	Training accuracy	Test Accuracy
SVM (83.07%)	γ=0.0001,C=0.001	86.746	85.256	87.163
Logistic regression (86.957%)	γ=0.01, σ^2=100	85.107	85.015	87.918
Decision Tree (88.242%)	Depth=7	88.829 (Best Tree from cross validation)	99.879	99.88
Bagged Forests (87.793%)	Depth=7	87.889	87.655	87.661
Aggressive Perceptron (87.953%)	Margin=0.1	86.358	86.76	88.175
Dynamic Learning Rate Perceptron (86.443%)	γ=1	81.38	86.75	86.391

On Kaggle, I put which csv file should be used for each algorithm. The following are the names of the CSV files that should be used including when I submitted them:

SVM:	”SVM.csv” (submitted Dec 7th 5:55PM)
Logistic Regression:	”Logistic_Regression.csv” (submitted Dec 6 7:19 PM)
Decision Tree:	”Best_Prediction.csv” (submitted Dec 7 5:09PM)
Bagged Forest:	”DebugBagged_Forest.csv” (submitted Dec 6 5:49 PM)
Aggressive Perceptron:	”Aggressive_Perceptron.csv” (submitted Oct 12 12:36 PM)
Perceptron with Dynamic Learning Rate:	”Dynamic_Learning_Rate.csv” (Oct 12 12:43 PM)

6 Continuing the Project

If I had more time to continue this project, I would definitely be able to add much more work to this project. First off, I would implement Naive Bayes just to make sure that it really does not give good accuracy for this project. Then I would implement Nearest Neighbor because it is a very easy algorithm to implement along with Naive Bayes and I would like to see if I could improve my accuracy. Once I had implemented those algorithms and most likely realize that they are not the best, I would continue to try to improve my Decision Tree by extracting more features from the dataset. I would probably try to add about ten more features from the raw data. After that I would take those ten additional features and add them to my Bagged Forest, and I would tune up my bagged forest and figure out how to get better accuracy. Finally, I would figure out why my bagged forest is predicting the same label for a single example, then I would be able to fix my SVM over trees and Logistic Regression over trees. Once that is complete, then I would explore even more and try all the perceptron algorithms over trees. After doing all this, I do believe that I would be able to increase my accuracy to an advanced level of Machine Learning.