Supervised Learning - Assignment 1
CS 7641 Machine Learning

Anishi Mehta (903390381)

September 25, 2018

1 Problem Definition

1.1 Sentiment Labelled Sentences

This is a dataset consisting of 3,000 examples of online reviews (taken from Amazon, IMDB and Yelp). Each example includes a binary label indicating whether the review is positive or negative. To use this example, we have to create a Bag of Words, and use the presence in a certain sentence as the features of that sentence. To perform well in this classification problem, the algorithm must learn the association that some words have with either positive or negative sentiments.

1.2 CIFAR-10

This is an image classification dataset. It is composed of 10 classes (airplanes, bird, cat, etc), with 10,000 images per class, each 32x32 pixels. Each pixel includes the red, green and blue channel values. The sheer size of this dataset, and the high dimensionality of the feature space makes this dataset an extremely challenging classification task. Doing well in it has proven to be very hard, which is why I believe this dataset is interesting. With this sort of a dataset, I hope to show performance differences between the different algorithms used.

1.3 Why These Datasets

These datasets were of interest to me, not only because they are hard, but more so because of their real-world applications. Performing object recognition in images is very useful and is used widely in the industry for a variety of purposes, like facial recognition, image tagging, etc. I am working on similar problems for my project at the Robotics Lab. On the other hand, identifying sentences and their sentiments is useful because it can help a review aggregator, for instance, separate praise and criticism. A lot of importance is now being given to Neural Language Processing, and I wanted to use this opportunity to explore the field a bit.

Thus, the intention was to use Machine Learning algorithms and apply them to a wide array of fields, demonstrating the power of these algorithms.

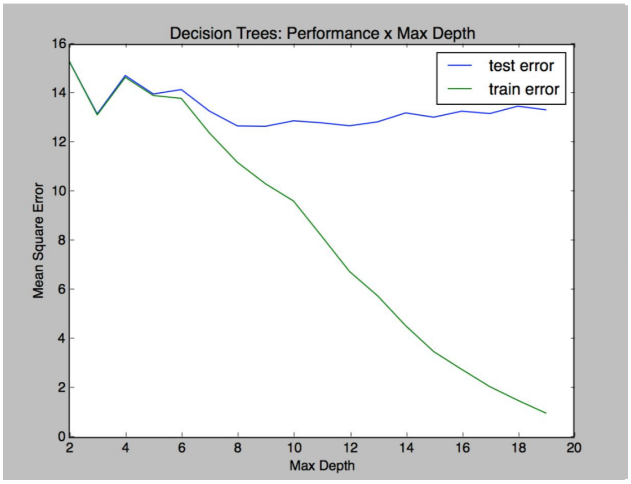

Figure 1: CIFAR: Performance of Decision Tree wrt. Max_depth

2 Running the Algorithms

Five different algorithms have been implemented on these datasets. These are - KNN, Neural Networks, Decision Trees, Boosting, SVM.

Each of the algorithms were trained over the training data multiple times. For each of them, multiple experiments were run to determine the most appropriate hyperparameters (e.g. how much to prune the decision tree, the number of hidden layers in the neural network, etc). The reason for these experiments was to find the set of hyperparameters that will generalize well, but still try and minimize overfitting. Once this was done, one last experiment was run to determine performance of the algorithm as a function of training set size. These results have been used to compare the performance of the different algorithms.

All the experiment runs were validated with the Mean Square Error of classification over the training and the test set. For both of datasets, the split between training and test set was fixed at 70% / 30%. For the experiments that train on different set sizes, the training data was taken as a percentage of the training set. In this case, the test set size remained constant.

Now, we look at each algorithm and analyze the results.

3 Pruned Decision Trees

A basic Decision Tree was created using sklearn. I used the GINI impurity coefficient to measure the quality of a split. The decision trees used were pruned by setting a max_depth. This is not an ideal way of pruning because it does not prevent Early Stopping, but I found that the performance and the generalization of the algorithm was still pretty good. However, it would be ideal to use some different form of pruning.

Figure 2: Sentiment: Performance of Decision Tree wrt. Max_depth

To begin with, we determine the max_depth of the decision tree i.e. how much to prune it. It is easy to see that with deeper trees, we get very low error rates on the training set, which is expected. What may not be very clear from the graphs (Figure:1, Figure:2) is that the test set error go slightly up as depth increases (this is more noticeable in the CIFAR-10 graph). This means that the tree is overfitting our training set. While this amount of overfitting could be considered negligible, I tried to minimize all overfitting, which had the added bonus of shorter training time. Eventually, the max_depths chosen were 8 for CIFAR and 35 for Sentiment.

Further, I tried to determine a good size for the Bag of Words in Sentiment. The results had very high variance, but consistently low in error, which showed that any bag size bigger than a certain number (initial bag size was 100) would perform well Figure:3. Just to be sure, I picked a bag size of 2000. The results can be seen in Figure:4 and Figure:5

This result turned out to be the case for all experiment runs in this paper.

As expected, as training size increases, we observe a decrease in train set accuracy i.e. increase in training error. With a bigger training set (and because were pruning) we will overfit the training set less, and get predictions that generalize better, which in turn explain the increase in test set accuracy i.e. decrease in error. What is interesting to note, however, is that even 10% of the training set is already a big enough set for predictions that generalize. We can see this in Figure:4 and Figure:5 by noticing the relatively small test errors for small training sets.

4 Boosted Decision Trees

For the boosted version of Decision Trees, I used AdaBoost, with a variable number of estimators as a starting point for this experiment. All runs had a fixed max_depth of 10.

Figure 3: Sentiment: Performance of Decision Tree with Bagging

Figure 4: CIFAR: Performance of Decision Tree with Bag Size 2000

Figure 5: Sentiment: Performance of Decision Tree with Bag Size 2000

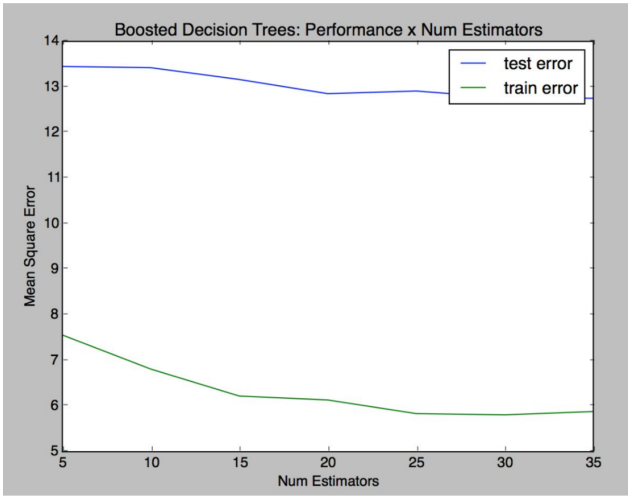

Figure 6: CIFAR: Performance of Boosted Decision Trees vs. Number of Estimators

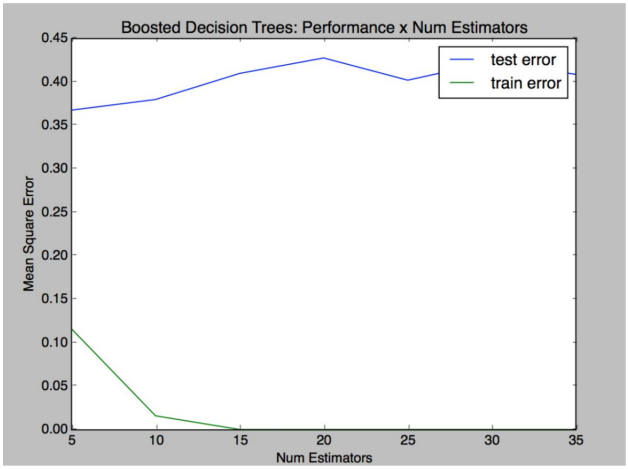

Figure 7: Sentiment: Performance of Boosted Decision Trees vs. Number of Estimators

As is observed from Figure:6 and Figure:7 in the graphs, for both datasets, any number of estimators greater than 10 provided a negligible increase in accuracy. For this reason, I chose 10 to be the number of estimators while running the variable training set experiment.

As can be seen from Figure:8 and Figure:9, the boosted version of our decision trees were capable of learning and classifying both test sets just as well as Decision Trees did. In both Decision Trees and AdaBoost, test error for CIFAR converged at around 13%, and for Sentiment at around 0.4%. The real difference AdaBoost made was in decreasing both training set errors by half. This shows that AdaBoost was successful in avoiding overfitting, by maintain small test set errors, and in learning generalized predictions, as it decreased the training set error overall.

To test this, I would like to have run one more experiment, with more data, to see if boosted DTs show an increase in performance over classic DTs, not only on the train set, but also

Figure 8: CIFAR: Performance of Boosted Decision Trees wrt. training set size

Figure 9: Sentiment: Performance of Boosted Decision Trees wrt. training set size

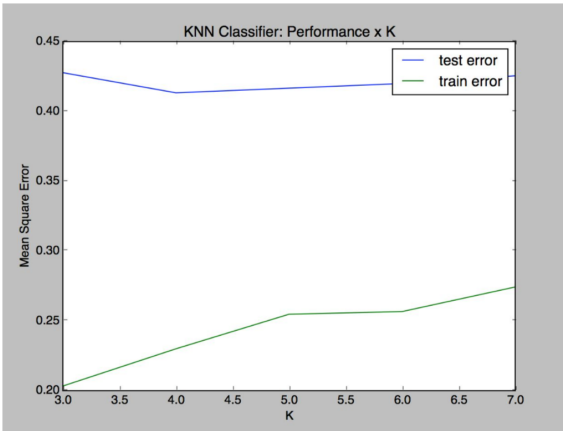

Figure 10: Sentiment: Performance of kNN with varying K

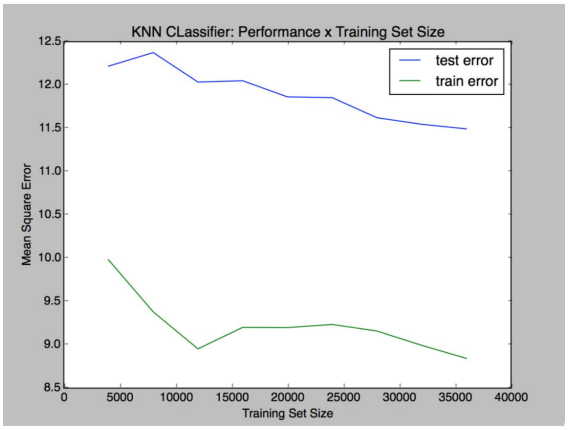

Figure 11: CIFAR: Performance of kNN wrt. training set size

on the test set. I believe this would be the case because, in the DTs learning curves, it is clear that the test error stabilized and is, in fact, starting to go back up again i.e. DTs are starting to overfit. However, this is not true for the learning curves in Figure:8 and Figure:9. Instead, the trend of the test error seems to be going down.

5 K Nearest Neighbor

For the kNN algorithm, the variable used to show performance as a function of complexity was K i.e. the number of neighbors used to make a prediction. Interestingly, this variable showed very little effect when training on the Sentiment dataset, as can be seen in the very small error variance shown in Figure:10.

Unfortunately, due to the complexity of the CIFAR dataset, a similar experiment could not be run, as it took too long to produce meaningful data. However, I believe the great performance of kNNs on it makes up for the lack of K-variation analysis.

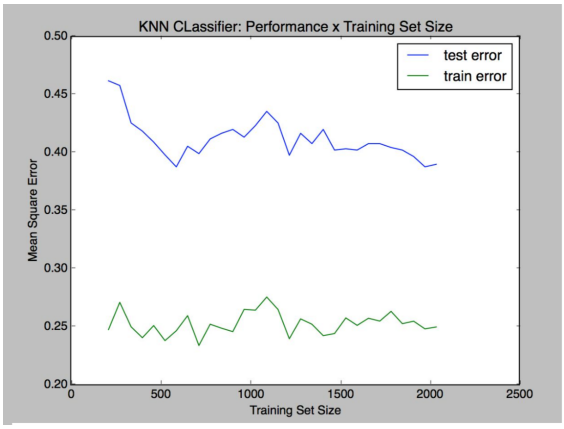

Figure 12: Sentiment: Performance of kNN wrt. training set size

As can be seen from Figure:11 and Figure:12, the kNN classifier performed extremely well on both datasets. On CIFAR, it got a slightly worse train error than Boosted Decision Trees, but it was able to reduce the test error below 12%, something which no algorithm has done so far. This means that the predictions it made were more generalized, even after the trade-off of getting a slightly worse performance on the training set. Moreover, as can be seen in Figure:11, both train and test error show a downward trend. This shows that this algorithm would benefit from more training data, and it would be intersting to see how it performs on bigger image classification datasets.

kNNs performed similarly well on Sentiment, but no significant differences are discernible when comparing this performance with Boosted Decision trees. This may speak to the lack of complexity in the Sentiment dataset rather than commenting on the algorithms themselves.

6 Support Vector Machines

SVM also took a long time to train. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples. However, the results obtained appeared to be a good trade-off for that.

Once again, varying complexity (in this case, it was the degree of the underlying function) did not affect performance on the Sentiment dataset. Hence, for both datasets, retained the default degree value of 3. As we see in Figure:13 and Figure:14, this choice performed very well, even better than kNNs.

It could have helped to spend more time to analyze performance on CIFAR with varying degrees of complexity. Perhaps better results could have been obtained had a different degree value been used.

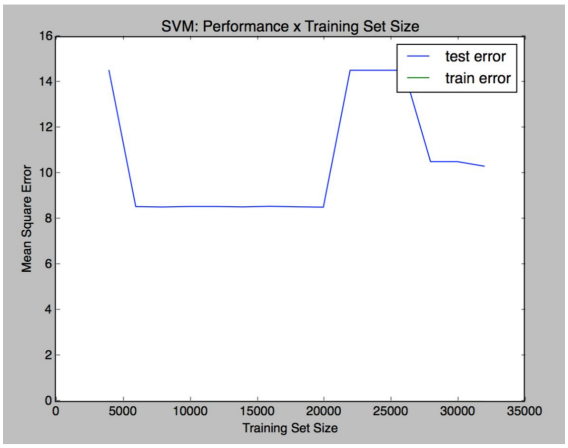

Figure 13: CIFAR: Performance of SVM with varying training set size

Figure 14: Sentiment: Performance of SVM with varying training set size

One thing to note from the figures right away is the highly sloped downward error trend in the Sentiment dataset (Figure:14). However, when we look at the scale on the y-axis, we see that while SVM seem to be performing better and better, it is actually trying to catch up to kNN. Another thing that stands out is the performance of SVM on CIFAR. It was able to achieve ¡ 9% errors on the test set consistently, which is lower than any we saw previously.

I see this as a fundamental difference in the working of the two algorithms. kNN is able to make very good predictions with much less data, as it only looks at K nearest neighbors at a time, while SVM seems to work better with larger amounts of data. Thus, in a dataset like Sentiment, which has a limited amount of data, SVM is still trying to match the performance of kNN. Meanwhile, on CIFAR, where an excess of data is available, SVM outperforms kNN by a considerable amount.

However, since SVM work better on larger amounts of data, it also suffers from a long training time. This is an interesting trade-off leading to questions which we face quite often in our applications. Given lots of data, do we contend ourselves with good data if it means

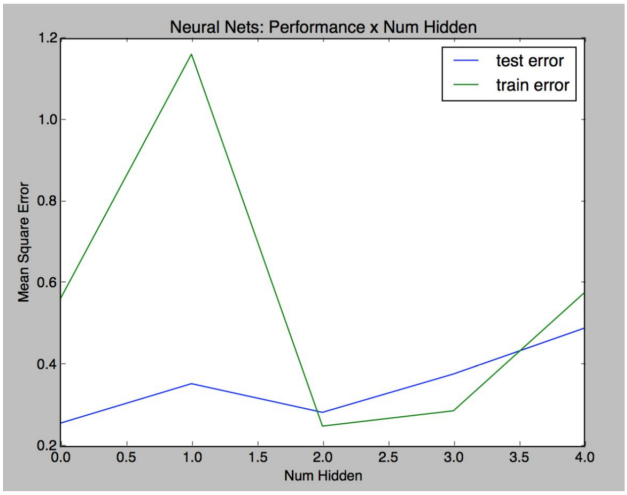

Figure 15: Sentiment: Performance by varying hidden layers

a shorter training time (if youre using kNN, a lazy learner)? Or should we spend a lot more time to get great results with SVM? The answer might be different according to the task we have at hand. For example, if we were looking at medical data to diagnose cancer patients, we would want to spend the time to make sure we have as accurate a classifier as possible.

7 Neural Networks

NN are among some of the most complex machine learning algorithms available today. However, they presented some interesting results. The Neural Networks were trained using a relatively small (but fixed) number of epochs, but the results were still impressive.

For the complexity-varying experiment, I varied the number of hidden layers being trained on the Sentiment dataset. As can be seen in Figure:15, having a certain number of hidden layers can make our NN produce a highly generalized classifier. However, if we increase n_hidden by too much and we get a neural network that is too complex to train on such a small dataset.

Thus, I decided to use 2 hidden layers for my Neural Networks experiments.

The first thing to point out about the performance of NN is the large variance observed between different runs in the Sentiment dataset (Figure:16). This is presumably due to the random initialization of weights in the NN, and the relatively small number of training epochs. However, I used Bagging to overcome the error of randomization through ensemble, and this further analysis showed something different than what was expected.

When we analyze the performance of NN with varying bag sizes (Figure:18), we see that more attributes cause the Curse of Dimensionality which is to say that NN is unable to learn which features matter, because it does not have enough training data to do so.

Thus, the main reason of its performance is the lack of Sentiment data. This is clear when we look at CIFAR, where data is abundant in Figure:16. NN achieved 2% test error, despite

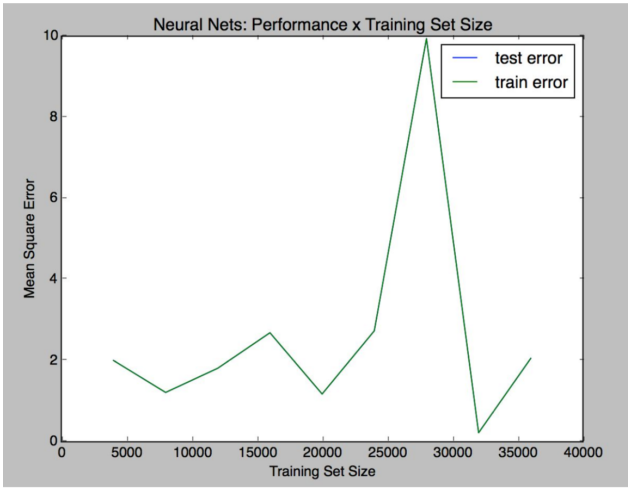

Figure 16: CIFAR: Performance of NN with 2 hidden layers

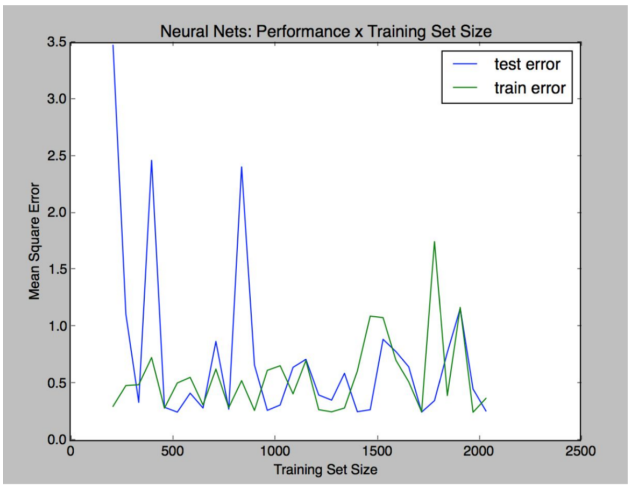

Figure 17: Sentiment: Performance of NN with 2 hidden layers

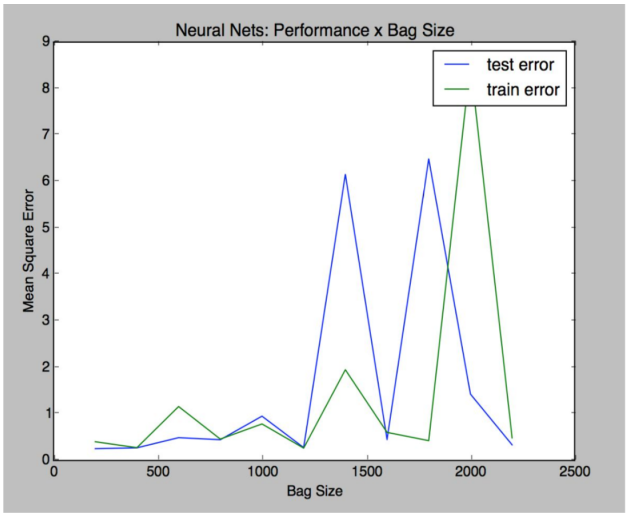

Figure 18: Sentiment: Performance of NN with varying bag size

the extremely high dimensionality of the attribute space.

Hence, I believe that given more Sentiment data, NN would be able to make near-perfect predictions, just like it did on CIFAR.

It is also important to note, however, that NN had a significantly large training time, with the cifar_setsize experiment taking over 40h to run on my laptop. While the training time was similar to that of SVM, NN significantly outperformed SVM. We could also record the training times, so that we could make a better analysis of the Time/Performance trade-off ratio.

8 Conclusion

In this paper, I hope to have shown a good comparison of various learning algorithms, and their performance in different datasets and conditions.

It is worth mentioning that all algorithms perform substantially better on the Sentiment than on CIFAR. The obvious reason for this is the Curse of Dimensionality. The CIFAR dataset has 10,336 features per example, against Sentiments about 2000 (depending on the size of Bag of Words). This implies that finding the features relevant for classification is much easier on Sentiment than on CIFAR. On CIFAR dataset, we could achieve much higher performance if we use something like Convolutional Neural Networks, which analyzes combinbation of features in a translation-invariant manner.

However, it is also important to note that the Sentiment dataset is a relatively simple classification task. The real interesting results occur when our algorithms achieve very low error rates on the CIFAR dataset, like Neural Networks did.

Using this, we have shown how some algorithms are capable of learning and generalizing better than others when theyre given more data and more features, while others perform better on less data and features. We have also shown the effects of this in training time (especially for eager learners) and how this trade-off needs to be kept in mind when choosing a classifier.