CIS 419 PS2

Varun S Rajagopal

February 28, 2018

Parameter Tuning:

1. Based on trials with the suggested parameters, for Winnow an alpha value of 1.005 was chosen as it led to the highest accuracy values when predicted on the development data (Syn Dense Dev Accuracy: 99.81, Syn Sparse Dev Accuracy: 97.97 and News Dev Accuracy: 80.82).
2. For AdaGrad, an eta value of 1.5 was chosen as it led to significantly higher values (by a very large margin) than all other parameter choices (final accuracy values of Syn Dense Dev Accuracy: 100.0, Syn Sparse Dev Accuracy: 89.1, News Dev Accuracy: 89.73).

Experiments with Synthetic/Real Data:

The following table summarizes the accuracy results on the development data after training each of our algorithms on the entire training set over the course of 20 iterations. Hence, these correspond to the last point in each of our learning curve diagrams. As can be seen in the table, over the course of the iterations and examples, all of our algorithms managed to perform at nearly a 100 percent accuracy on the development data, with the lowest value being 99.94 with the **Averaged Winnow** algorithm.

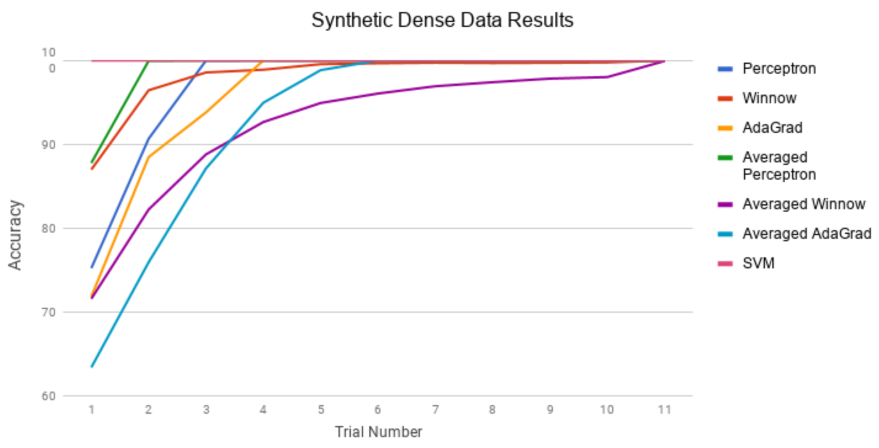
Final Dev Accuracy Values With 20 Iterations:

Algorithm	**Synthetic Dense Dev Accuracy**	**Synthetic Sparse Dev Accuracy**
Perceptron	99.99	99.99
Winnow	99.99	99.9
AdaGrad	100	100
Averaged Perceptron	100	100
Averaged Winnow	99.94	99.76
Averaged AdaGrad	100	100
SVM	100	100

The following table shows the results of our two chosen algorithms, **averaged basic perceptron** and **SVM** on both the synthetic and real data sets. As can be observed, **SVM** outperformed averaged perceptron by a small margin. Interestingly enough, despite the fact that the email dataset is from an entirely different database, the trained models from our news data generalized well and gave us high accuracy values with the enron dataset.

Data	**Averaged Perceptron Accuracy**	**SVM Accuracy**
Conll	91.06	92.99
Enron	89.28	91.70
Synthetic Dense	100	100
Synthetic Sparse	100	100

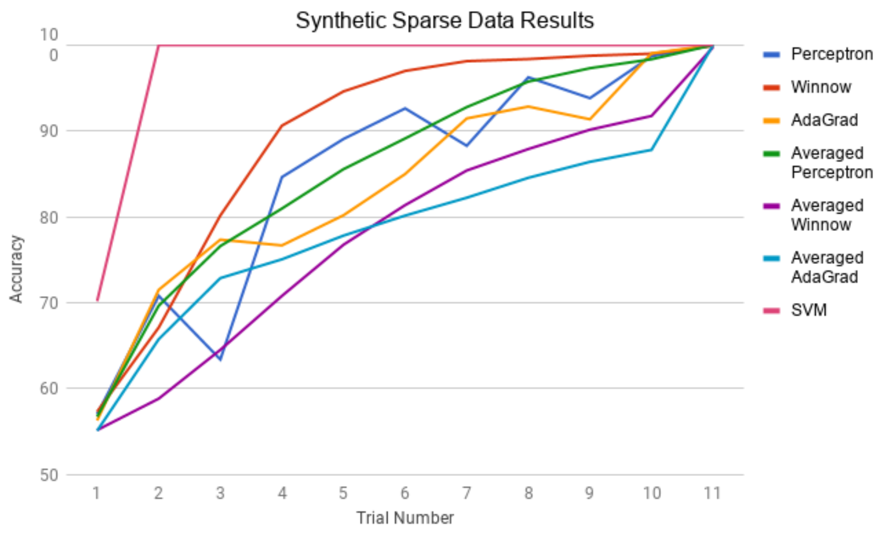
Learning Curves For Synthetic Dense Data:



The above diagram with the attached legend depicts all of our algorithms throughout the learning process with each of the color coded plots representing an individual learning curve. As can be seen in the accuracy values corresponding to trial number 11, all of the algorithms reached nearly a 100 percent accuracy at the conclusion of our experiment. In comparing these algorithms, there are a few results that stand out.

Firstly, we can see quite clearly that the best performing choice was **SVM** as it consistently had an accuracy of 100 percent over the course of our trials, regardless of the size of our training set. Additionally, by trial 6 all of our algorithms except for **Averaged Winnow** had converged to a 100 percent accuracy and this is probably due to an unfavorable initialization of parameters with the algorithm. A clear takeaway from the diagram though is that our choice of **Averaged Basic Perceptron** was good as it required fewer unique training examples to reach a 100 percent accuracy than its counterpart, the non-averaged version. The reason for this is likely to be that assuming that draws of data are i.i.d, by giving higher weight to the models that survive longer, we maintain a better weight matrix. Moreover, **Averaged Basic Perceptron** also outperformed **Winnow** and **AdaGrad** as it has a higher accuracy for all trials and this is probably due to to the same reason highlighted above. Moreover, the fact that Winnow does not update theta and that AdaGrad frequently changes the learning rate might also be reasons that the two algorithms reach higher accuracy values slower than Average Perceptron. In the case of **AdaGrad**, the changing learning rate that can slow down convergence is also likely a good indication of why the **Basic Perceptron** algorithm outperformed AdaGrad overall. A rather surprising result though is that **Averaged AdaGrad** was one of the slowest to reach an accuracy value above 99 percent and this might be due to the fact that we are working with a dense dataset. Since several features could be changing often in a dense dataset, the smaller learning rates given by AdaGrad might mean that weights do not get updated in an extreme enough manner to facilitate avoiding making mistakes on similar examples in the future. Finally, in comparing the basic versions of the three perceptron algorithms, they seem to all converge to high accuracies at similar rates. This is despite **Winnow** starting at a higher accuracy on the lowest number of training examples (likely due to its good initialization parameters through parameter tuning). Finally, one possible explanation for why **Averaged Winnow** and **Averaged AdaGrad** performed worse than their basic counterparts might be a result of using the same alpha and eta parameters that were chosen to produce the highest accuracy values in their respective non-averaged cases.

Learning Curves For Synthetic Sparse Data:



As in the previous diagram, each of the learning curves is represented in a different color as indicated in the legend. Overall, we observe that with a sparse feature representation, all of the algorithms (barring **SVM**) require more training examples to reach a higher accuracy level. This is intuitive and follows as a general problem of a sparse feature space as described in lecture. We also see that most of the algorithms start at around 60 percent accuracy and then slowly begin to increase until they finally all reach close to a 100 percent accuracy when given the entire training set. Overall, the growth in accuracy in these learning curves looks to be more linear in the size of the example set as compared to before, but with higher variance in the rates of improvement in the algorithms (given an increase in training examples) when compared to one another.

The only difference in the sparse case with **SVM** is that it has lower accuracy on the 500 example training set but with 20 iterations reaches a 100 percent accuracy from a 1000 examples onwards. **AgaGrad** performs better in comparison to Basic Perceptron as compared to in the dense case likely due to the fact that the variable learning rates creates a larger difference when the data is sparse. The **AdaGrad** performance is similar to the **Averaged AdaGrad** performance which is possibly a result of the fact that the more updates made to Adagrad, the better the model is compared to previous iterations. This also likely holds true when compared to the disparity between **Averaged Perceptron** and **Basic Perceptron** (since the dynamic learning rate in **Adagrad** is better specified after looking at more examples), hence indicating that weighting the previous examples negates the benefit of averaging. Surprisingly though, **Averaged Perceptron** and **Regular Perceptron** perform similarly to one another on each of the training sets (with the averaged version having a slight advantage) and this may be due to the fact that learning on sparse data does not significantly change the weight matrix with every example. However, as the number of training examples (m) increases, **Perceptron** dips and grows in accuracy while the averaged version steadily increases and this is likely a function of the small number of examples not satisying our general i.i.d assumption. Finally, **Winnow** performs well on the sparse dataset in comparison to the other Perceptron algorithms and this is probably a result of the multiplicative scheme that it is biggest advantage. Since there are a large number of irrelevant features, Winnow's updating mechanism produces better results.