Introduction

Group Members:

Eli Pinkus
Akshay Vegesna

Group Name

Yasser's Disciples

Division of Labour

Each of us implemented models on or own and we generally discussed the merits of models and what was or wasn't working as a group. Labour was well divided, and although some group members models performed better than others, the overall work load was well distributed.

Overview

Models & Techniques

- **Ridge Regression:** This simple model performed poorly out of sample, with an average 8-fold cross validation error of 0.845 percent.
- **Lasso Regression:** This model also performed poorly in general, and the cross-validation error plateaued at around 0.845 percent. The sparse weights were used as a method of dimensionality reduction.
- **Logistic Regression:** This model performed very well—much better than Ridge or Lasso. This model ended up performing the best out of sample out of all of our other models. A grid-search was done with 6-fold cross validation to find the best regularization hyper-parameters.
- **SVM (Linear):** This model was difficult to run because the input space was so big. We used the scipy.sparse package to speed this up because we understood that most weights were zero. After this key step, our SVM model performed well, with a training error of 0.88 but a validation error of around 0.85. We attempted to improve this model by tuning the regularization parameter, but changing the regularization parameter did not improve the performance of the SVM significantly.
- **Shallow Neural Network:** Performed slightly worse than standard logistic regression. Noticeable problems with overfitting.
- **Decision Tree with Gini Loss:** Poor performance despite tuning of regularization paramters.
- **Random Forests:** This approach did not perform much better at all than Linear SVMs, and led to 0.83 training loss.
- **Adaboost:** We used Adaboost with Decision Trees, but the validation errors we obtained were only around 0.84. We changed our approach and used Adaboost with Logistic Regression and got a consistent validation error of 0.848, but were not able to improve upon this. We submitted a prediction as well, but got 0.847 error on the test set.

- **Feature Reduction:** We tried to reduce the dimensionality of the input space based on various criterion with mixed results.
- **K-Fold cross validation:** We used K-Fold cross validation to get a sense of how a given model was performing on unseen data. In the many cases were the running times were too long, we used simple validation sets instead of K-fold cross validation.

Work Timeline

- **Week 1:** We ended up doing most of the pre-processing and implemented some basic linear models to understand the dataset. We also discussed and implemented various methods of feature reduction because we foresaw that the noise from the bag-of-words input space would make the classification task difficult
- **Week 2:** We implemented most of the well-performing classifiers we obtained during this period, and achieved the best results with Logistic Regression. Our key idea then was to use Neural networks with a sigmoid activation function to get more sophisticated decision boundaries, though we were unable to improve upon basic Logistic Regression out-of-sample.

Approach

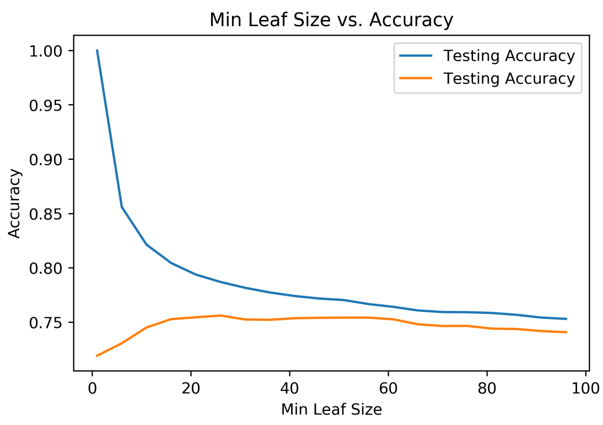
Data Processing/Manipulation

The bulk of our data manipulation focused on feature selection with views toward reducing the dimensionality of the input space. We began by considering a restriction on the minimum letters in a word represented in the 'bag of words'. We do so under the rough assumption that longer words would, in general, be more specifically employed, and thus, more meaningful in sentiment analysis. We also implemented more principled approaches to feature reduction and we implemented lasso regression and principle component analysis for certain models as will be discussed more in-depth in discussing the implementation of each model. In general, we looked to apply these techniques more aggressively with more complicated models that have high variance in an attempt to reduce the complexity of those models and drive down variance. We used the SciKit Learn implementation of PCA. Unfortunately, these efforts didn't seem to pay high dividends in performance in neither complex models nor simple models. Our overall best classifier (SKLearn logistic regression), performed best with the full 1000 feature input space. This was a surprise to us because we were confident that there were at least several words in the 'bag' that were meaningless in sentiment analysis.

Details of Models and Techniques

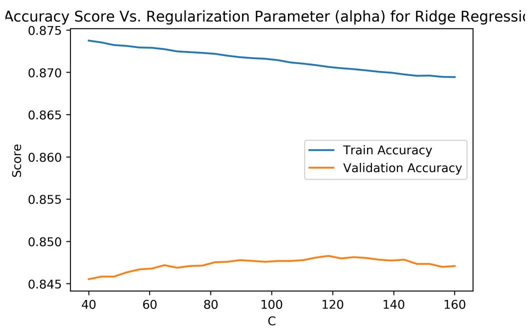
Our first effort was implementing basic linear models to give us a baseline for performance and hopefully motivate selection of more advanced models.

Decision Trees and Adaboost: We first used decision trees with gini impurity, and found that validation error plateaued at around 0.75, when we regularized using minimum leaf size.



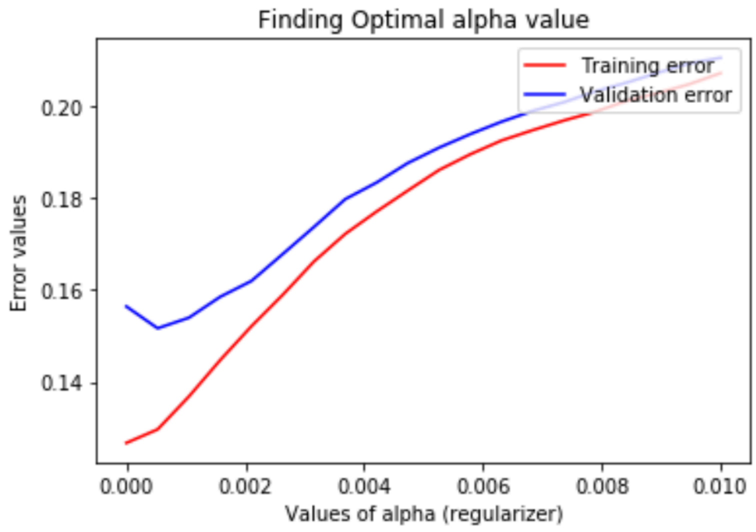
While we were not very excited by the performance of decision trees, we implemented them with Adaboost to check whether this would be successful. It turned out to not be successful and we obtained values of approximately 0.825 for our training error and 0.82 for our testing error. We tried searching for better "weak learners" but were quickly unable to find any.

L2 Regularized least squares regression: We began by implemented L2 regularized least squares regression with hard thresholding for classification. We did so using the Ridge package from SKlearn.linear_models. We experimented with the regularization parameter, alpha, in effort to find a point of acceptable bias-variance compromise (see figure):

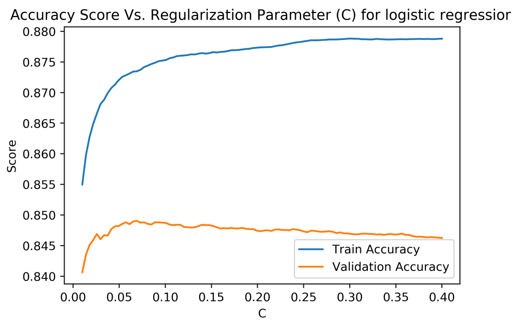


As expected our performance with this simplistic model was not competitive as our best validation accuracy was noticeably lower than 0.85. It was advantageous because it was simple and executed quickly, however it performed poorly.

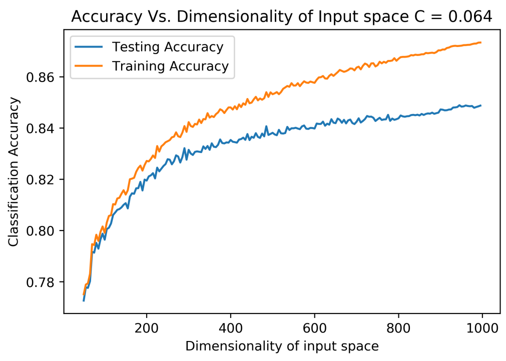
L1 Regularized Least Squares Regression: We again implemented Lasso from sklearn.linear_models. This model gave surprisingly great performance. After we did a grid-search for regularization value, we found one that gave the best performance consistently on validation sets (around 0.848). Interestingly when we checked the values of the weights at this model, it turned out that 300 weights were sparse, and we took this to mean that these 300 weights may not be contributing to the efficiency of the model. We created a new training and test sets without these 300 weights, and used it to train on for most of the other models to try and improve performance. However, this was largely unsuccessful and we could not really improve on performance by removing sparse weights. Variants of this approach were also attempted, by removing 100 and 50 features, but with no better performance out of sample. Below is a figure of finding the validation error for Lasso classification.



Logistic Regression: We implemented logistic regression using sklearn. Initial efforts with the model showed strong performance and after attempts with more complicated model classes, we came back to logistic regression as our best performer. We began by searching for the best regularization parameter, C, which we found to be around 0.06433. (see figure):



Dimensionality Reduction and Logistic Regression: Encouraged by our initial findings with the model, we looked to apply principle component analysis as a pre-processing tool to feed into logistic regression. As mentioned previously, our hope was that we could reduce the dimensionality of the input space without losing too much relevant information. This technique did not help with logistic regression, as reduction of dimension resulted in decreasing performance as illustrated in the following figure which we obtained by fixing the regularization parameter and varying the dimensionality of the input space produced by PCA:



Unfortunately, these efforts did not yield performance improvements, however we were still satisfied with the performance of logistic regression on the full, 1000-dimensional sparse input space.

Model Selection

Scoring

For the most part we stuck to using binary classification error on validation sets to get intuition about which of our models were performing better. The lowest scores were probably when we used SVM's with an RBF kernel (around 60%) and the highest consistent scores that we got were around 85% with simple logistic regression.

Validation and Test

We used cross-validation in most cases to choose the best parameters for a classifier. We had a simple python script for a grid-search of regularization parameter which we just ran with every model and checked whether performance was improved.

While cross-validation was a great idea for most models, we encountered other cases like linear SVMs for which run-time made cross-validation unfeasible. Simple validation sets were used instead in these cases, though we understood that the values we were obtaining were far less reliable.

Conclusion

Discoveries

We learned that sometimes simple linear models can still beat well-tuned complex models in certain situations. This is likely to do the relatively low variance in this model class compared to models with more degrees of freedom. We believe that our use of a simple model was at least partially responsible for our rank increase from the public leader board to the private leader board as we find it likely that many groups implemented more complicated, high variance, models and as a result had the tendency to over fit to the public leader board data as they sought high performance in that domain.

We also have reason to believe that logistic regression is a useful model in cases, such as this one, where input space is high dimensional and sparse.

Challenges

We were unable to find reliable ways to extract relevant features from the bag-of-words input data. Each method we used to reduce feature space was outperformed by simple logistic regression on the original features. This pointed to the fact that our methods to reduce dimensionality were not effective, though we did end up spending a lot of time on them. The bag-of-words input model made it extremely difficult to deal with the data.

Concluding Remarks

We were somewhat unsatisfied with the fact that our final model was the model we found so early on. However, we were satisfied in that we tried almost all other models that we learned in the course and believe that we understood a great deal about their effectiveness for classification tasks. Given additional time, we would probably test out classification learning models that were recommended online for bag-of-words datasets.