Introduction

Group members
Anne Chen, Jenny Lee, and Jerry Feng

Team name
THE MAGIC CONCH

Division of labour
All three members participated in an initial identification and coarse validation of methodology. Once the top model candidates were decided, they were split among the team members for further exploration. Anne worked on Logistic Regression models. Jerry worked on Naive Bayes and SVM models. Jenny worked on Neural networks.

Project Link:
https://github.com/clee7/Sentiment-Analysis-of-Amazon-Reviews

Overview

In this project, we performed sentiment analysis on a bag of words representation of Amazon review data. The team started by researching more about the general topic and looking into different possible machine learning classifiers/methods.

Models and Techniques Tried:

We looked into the different classifiers available through scikit learn, as well as neural and convolution networks using the keras package. We proceeded to tune the parameters and use different methods of bagging and ensembling to optimize the performance of the best classifiers. The final validation errors of each was gathered using five fold cross validation.

Here's a summary of the results:

- In some of the classifiers, there were clear signs of overfitting - the training validation was high (~0.90 - 0.99) but the resulting testing accuracy was much lower (~ 0.7). We remedied this by implementing various regularization and looking into classifiers that did not overfit.
- After testing with all of the classifiers, we saw that Logistic Regression models, SVM models and Neural Networks had the lowest validation error, while Linear Regression had the worst.
- We further tried to improve test accuracy by bagging/ensembling layers of the best performing classifiers and generating a prediction with it. This proved to have a significant effect on the performance, and boosted our test accuracy above the targeted 0.85%.

Work Timeline:

- For the first week of the project, the team worked to understand the task and to implement a basic functional version of all the classifiers. We used the highest scoring classifier to generate a prediction for submission.
- In the second week, we worked to tune the parameters and optimize test accuracy. We also looked into different methods of data preprocessing.

Approach

1. Determination of models and techniques

No initial assumptions were made about the performance of the different classifiers. Instead, the group began with a general audit of potential methodologies. We coded up skeleton methods for each and tested it against training data by using five-fold cross validation.

2. Model Selection

- **Linear Regression**
  - Scikit learn's LinearRegression model was used. As one might expect, it was inefficient for our purposes and yielded approximately 50% accuracy on cross validation checks.
- **Logistic Regression**
  - Scikit learn's LogisticRegression model was also used. Logistic Regression is a classifier that is commonly used when there is one dependent variable and one/multiple independent variables.
  - Out of the gate, Logistic Regression was one of our strongest performers, able to easily reach 84% accuracy.
  - We later tried to improve test accuracy varying the regularization parameters and by using LogisticRegressionCV, which reached a slightly higher accuracy than LogisticRegression.
- **Decision Trees**
  - We used the DecisionTreeClassifier from scikit learn's tree package. A decision tree splits the data into 'branches' based on different characteristics and criteria.
  - Contrary to our expectations, it proved to have a lower test accuracy than other models possibly due to high variance and sparse feature vector of the bag-of-words representation.
- **Random Forests**
  - We also used the RandomForestClassifier from scikit learn's ensemble package. The Random Forest Classifier operates by constructing a multitude of decision trees and uses averaging to lower overfitting.
  - Like the Decision Tree classifier, the Random Forest classifier proved to have lower accuracy than many of the other models tested.

- **Support Vector Machines**
    - Support vector machines classify the data using a set of supervised learning algorithms. By providing it with labelled training data, the SVM returns a hyperplane that can then be used to classify new data points.
    - We primarily used LinearSVC, due to its quick runtime. NuSVM was also attempted. Though we were unable to finish running a gridsearch to conclusively rule the method out, we decided that a better use of time would be to shift our focus to improving the performance of the Neural Network.
- **Adaboost**
    - We used the default estimator (decision stumps). We expected Adaboost to perform similarly to random forest for this dataset, which it did.
- **Naive Bayes**
    - Naive Bayes is a family of categorization methods that assume that there the feature dimensions are strongly independent. Intuitively, Naive Bayes would appear well suited for our task, as the presence of a word in a document usually has no bearing on whether another word appears in that same document. Furthermore, Naive Bayes has been used for text-classification tasks, such as spam filtering.
    - Both the Multinomial and Bernoulli Naive Bayes methods yielded decent cross validation accuracy. However, grid-searching on their hyperparameters yielded no improved performance, and Naive Bayes eventually was overshadowed by our Neural Network implementation.
- **Neural Networks**
    - Neural networks are able to learn nonlinear features by using deep networks to classify the data. However, neural networks output complex models and can lead to overfitting. To avoid overfitting, number of epochs is lowered as well as the number of layers and neurons in our neural network.
    - Further investigation finds that using the optimizer ‘adam’ increases validation accuracy in comparison to the optimizer ‘RMSprop’.
    - As expected, it proved to be very effective in classifying the data and returned one of the highest accuracy rates after some parameter tweaking.

Upon testing, Logistic Regression, SVM, and Neural Networks seemed most promising, each capable of scoring in the 83-84% accuracy range with little tweaking when applying five-fold cross validation. Though it generally scored lower than the other three models, Naive Bayes was added in as well due to its historic use with classifying documents (such as spam filtering).

We continued trying to optimize the test accuracy rate by looking and methods of data preprocessing, bagging and ensembling.

3. <u>Data Processing And Manipulation</u>

- **Data preprocessing**
  The data that we received was already processed into a Bag-of-Words format. The processing tasks we applied did merely reweighted the existing information.

- *Count*: A simple count for how many times a term appears in the document
- *Frequency*: The count divided by document length. Because we did not have the full document, the document length was the L1 norm of the input data.
- *Binary*: 1 for if the document contains the term and 0 otherwise.
- *Term frequency–inverse document frequency*: This heuristic aims to give terms that appear in many documents a lesser weight than those that appear in few documents. In general, the weight for a given term t in input data x is:
  `tfidf(x, t) = tf(x, t) * idf(x, t, N)`

  Let `f(x, t)` yield the number of times a term t appears in a given document x, and `g(x)` return the number of times a term appears at least once in a document over the whole corpus. In our implementation,

  ```
  tf(x, t)     = log(1 + f(t, x)), and
  idf(x, t, N) = log(N / (1 + g(x))
  ```

- **Ensembling**
  - Ensembling is the technique of creating multiple models, generating predictions based off of them, and then combining them into one optimal prediction. We attempted to use this technique to decrease variance and bias of the final prediction.
  - We used a total of five models: Neural Network, Convolution Network, Bernoulli Naive Bayes, Logistic Regression, and Support Vector Machines. We trained these 5 classifiers using the training data set and combine the final prediction using majority vote.
  - When we applied this technique, we see that our validation accuracy (and test accuracy) increased slightly and passed the targeted 85%.

- **Bootstrap Aggregating ('Bagging')**
  - Bagging is an ensemble method that allows us to generate multiple data sets by picking randomly with replacement from the original training dataset. The results generated are then averaged together to create a prediction.
  - This method is highly useful when applied to classifiers that have high variance - since results from individual training sets are averaged, overfitting on one instance does not have as drastic of an effect on the final prediction than it would otherwise.
  - When we applied bagging to our Neural Network classifiers, it reduced the effect of overfitting and raised the test accuracy.

Conclusion

1. <u>Results</u>

We received the following training and test accuracies for the respective classifiers used:

```
================================================================================
CROSS VALIDATION: ClassifyWithNeuralNetwork
================================================================================
average training accuracy 0.9098625000000002
average val accuracy 0.8493999999999999
================================================================================
CROSS VALIDATION: ClassifyWithConvolutionNetwork
================================================================================
average training accuracy 0.9508625
average val accuracy 0.8426500000000001
================================================================================
CROSS VALIDATION: ClassifyWithLinearRegression
================================================================================
average training accuracy 0.0
average val accuracy 0.4527801646658468

================================================================================
CROSS VALIDATION: ClassifyWithRandomForest
================================================================================
average training accuracy 0.99185
average val accuracy 0.7821
================================================================================
CROSS VALIDATION: ClassifyWithSVM
================================================================================
average training accuracy 0.8761375000000001
average val accuracy 0.8477
================================================================================
CROSS VALIDATION: ClassifyWithSVC
================================================================================
average training accuracy 0.5714875
average val accuracy 0.56575
================================================================================
CROSS VALIDATION: ClassifyWithBernoulliNB
================================================================================
average training accuracy 0.8253250000000001
average val accuracy 0.817
================================================================================
CROSS VALIDATION: ClassifyWithMultinomialNB
================================================================================
average training accuracy 0.8391
average val accuracy 0.8307

================================================================================
CROSS VALIDATION: ClassifyWithLogisticRegression
================================================================================
average training accuracy 0.8786875000000001
average val accuracy 0.8455
================================================================================
CROSS VALIDATION: ClassifyWithAdaBoost
================================================================================
average training accuracy 0.8001375
average val accuracy 0.79445
================================================================================
CROSS VALIDATION: ClassifyWithDecisionTree
================================================================================
average training accuracy 0.9999500000000001
average val accuracy 0.7201000000000001
```

```
================================================================================
CROSS VALIDATION: Ensemble
================================================================================
average training accuracy 0.888305
average val accuracy 0.8503000000000001
================================================================================
CROSS VALIDATION: Bagging
================================================================================
average training accuracy 0.9103474999999999
average val accuracy 0.8524499999999999
```

After careful analysis of the results obtained from these different classifiers, we came to the conclusion that it was best to use bagging with neural networks. This combination results in the highest test accuracy consistently.

2. Discoveries

One major discovery we made as a team was that, in general, our prediction accuracy was dictated by the quality of the data. Continued improvements in performance in the top models became marginal and towards the end we were spending copious amounts of time trying to improve cross validation scores by as little as 0.1%. However, our research into existing literature implied that similar sentiment analysis tasks could reach 90% - 95% accuracy depending on the preprocessing approach. In particular, it seemed that n-gram approaches did well, whereas we were given our data in a 1-gram format.

Another surprising discovery to us as a group was the power behind statistical prediction models. Right out of the box, many techniques such as Naive Bayes and Linear Regression were able to reach 80% accuracy. However, meta-algorithms such as ensembling can further improve the test accuracy of our models by either reducing bias or variance.

3. Challenges

The data provided was preprocessed, which made it easier to obtain and work with. At the same time, this also limited the ability to run many different data processing methods on it that might have significantly boosted the test accuracy levels.

References

[1]https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a?gi=8d91a46f390

[2]https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[3]https://www.quora.com/What-are-the-best-supervised-learning-algorithms-for-sentiment-analysis-in-text-I-want-to-do-text-analysis-to-determine-whether-a-news-item-is-positive-or-negative-for-a-given-subject

[4]https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6

[5] https://machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis/