CS 5350/6350: MACHINE LEARNING FALL 2018

FINAL REPORT:
CLASSIFYING MOVIE REVIEWS

Prince Osei Aboagye

School of Computing
University Of Utah

December 15, 2018

1 Introduction

Sentiment analysis/classification is a common application of Natural Language Processing (NLP) methodologies where text is understood and the underlying intent is predicted. Sentiment classification has a lot of applications in the field of business intelligence, recommender systems, online surveys, message filtering, assessing movie review polarity, among others. The goal of this project is to predict whether a movie review is a positive or a negative one.
The data used for this task is based on the Large Movie Review Dataset v1.0 [2]. Each review in this task is characterized by the histogram of the words it contains. The labeled data set consists 25000 training examples. There are 12500 test and development examples.
In this report I will outline the results I obtained from the following learning algorithm: Simple Perceptron, Average Perceptron, Support Vector Machines, Logistic Regression, Bagging on Support Vector Machines, AdaBoost. For the first five learning algorithms (Simple Perceptron, Average Perceptron, Support Vector Machines, Logistic Regression, Bagging on Support Vector Machines) I used the extracted features that was given to us. However I went ahead to also extract my features from the scratch and then I applied it to AdaBoost.

2 Text Pre-processing

First of all, we load our movie reviews data which consist of training, testing and development data into python. The movie reviews data is read into python in the form of sentence token. This is then followed by text normalization. After sentence tokenization various other techniques including cleaning text, case conversion, expanding contractions, correcting spellings, correcting repeating characters, removing stopwords and other unnecessary terms, stemming, and lemmatization were performed in other to get the textual data into a form that can be easily understood and interpreted for our classification task.

3 Feature Extraction

Feature extraction is a process whereby we extract meaningful features or attributes from raw textual data for feeding it into a statistical or ML algorithm. I used a generic function to perform various types of feature extraction from the movie reviews data. The types of features I worked with are as follows: Binary term occurrence-based features, Frequency bag of words-based features and TF-IDF-weighted features. I used the sklearn.feature_extraction module to help me accomplish this task. From the sklearn.feature_extraction module I imported CountVectorizer, TfidfVectorizer and then used them in my feature extractor function. After applying the feature extractor function to the movie reviews data which consist of training, testing and development data I was able to extract 67293 features.

4 Model Training and Evaluation

For these four implementations (Simple Perceptron, Average Perceptron, Support Vector Machines, Logistic Regression) I run a 1-fold cross validation for ten epochs for each hyper-parameter combination to get the best hyper-parameter setting.
Then I train the classifier for 20 epochs. At the end of each training epoch, I measure the accuracy of the classifier on the development/validation set. I split the training set into a ratio of 70:30 in other to get a development dateset except for AdaBoost where I used the entire training set.
Also, I used the classifier from the epoch where the development set accuracy was highest to evaluate on the test set.
In this report I will outile the following:

1. The best hyper-parameters
2. The accuracy for the best hyperparameter obtained during cross-validation
3. Development set measurement accuracy for each of the 20 epochs during training.
4. Test set measurement accuracies - accuracy, precision, recall and F1 score
5. A learning curve where the x axis is the epoch id and the y axis is the development set accuracy using the classifier at the end of that epoch.

4.1 Simple Perceptron

The table below is a summary of the results on Simple Perceptron:

Best Hyper-Parameter	Accerage Cross Validation Accuracy	Test Set Measurement Accuracy	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate η: 0.1	87.133%	Best accuracy on 17th epoch: 87.413%	Accuracy: 86% Precision: 87% Recall: 86% F1 Score: 86%	86%

4.2 Average Perceptron

The table below is a summary of the results on Average Perceptron:

Best Hyper-Parameter	Accerage Cross Validation Accuracy	Test Set Measurement Accuracy	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate η: 0.01	88.173%	Best accuracy on 20th epoch: 88.267%	Accuracy: 87% Precision: 88% Recall: 87% F1 Score: 88%	87.152%

4.3 Support Vector Machines

The table below is a summary of the results on Support Vector Machines:

Best Hyper-Parameter	Accerage Cross Validation Accuracy	Test Set Measurement Accuracy	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate γ: 1e-05 Regularization Parameter, C: 100	87.64%	Best accuracy on 14th epoch: 88.667%	Accuracy: 87% Precision: 89% Recall: 86% F1 Score: 87%	87.312%

4.4 Logistic Regression

The table below is a summary of the results on Logistic Regression:

Best Hyper-Parameter	Accerage Cross Validation Accuracy	Test Set Measurement Accuracy	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate γ: 0.001 Tradeoff σ^2 : 1000	86.453%	Best accuracy on 19th epoch: 86.64%	Accuracy: 86% Precision: 86% Recall: 87% F1 Score: 87%	86%

4.5 Bagging on Support Vector Machines

For this implementation I created a multitude of 10 datasets of the same length as the original training dataset drawn from the original dataset with replacement . I then train 10 SVM model for each of the bootstrapped datasets and take the majority prediction of these models for a unseen query instance as our prediction. I used the best hyper-parameters obtained from our original SVM model for training and I run it for 20 epoch for of the 10 SVM model.

The table below is a summary of the results on Bagging on Support Vector Machines:

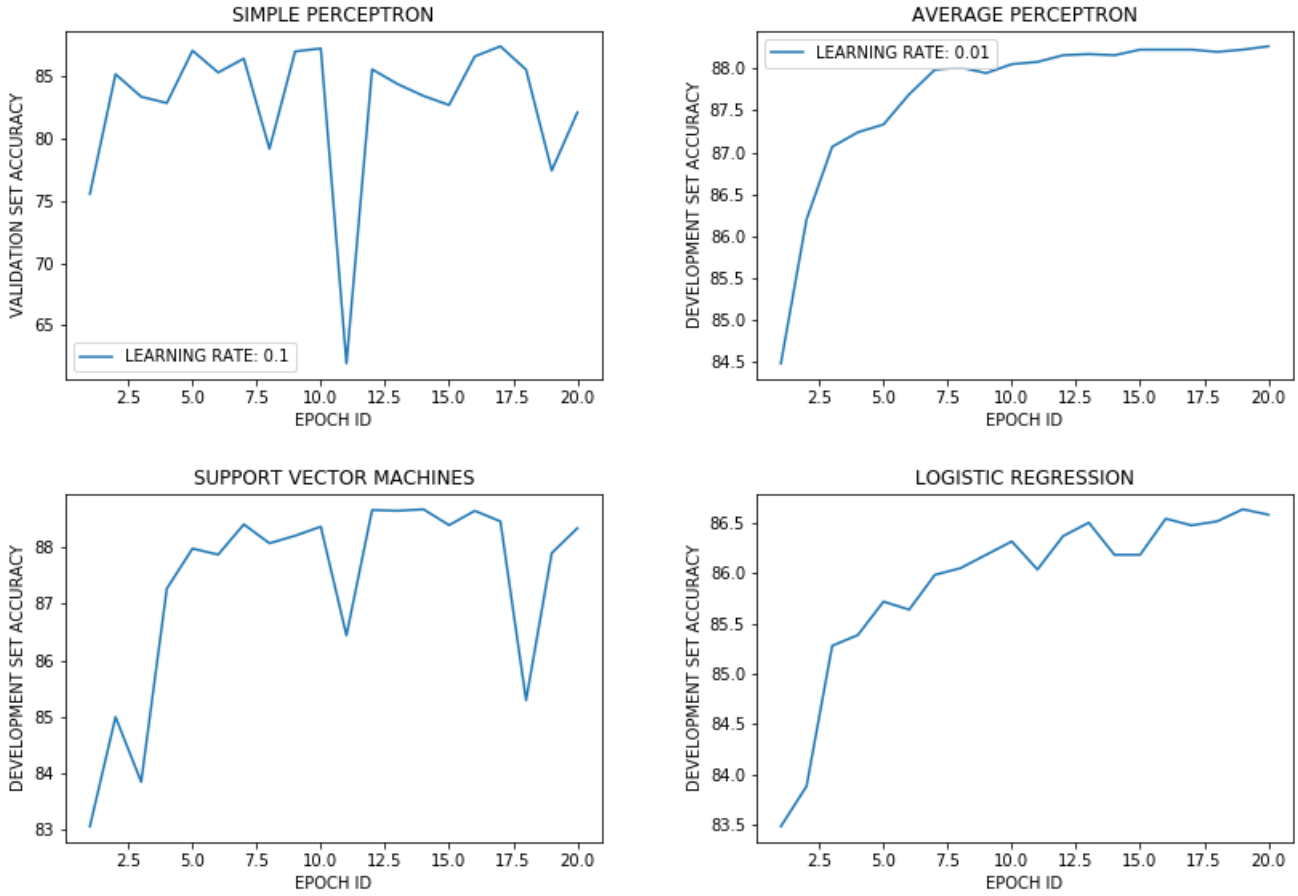
Best Hyper-Parameter	Accerage Cross Validation Accuracy	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate γ: 1e-05 Regularization Parameter, C: 100	87.64%	Accuracy: 87.68% Precision: 89% Recall: 86% F1 Score: 88%	86.912%

4.6 AdaBoost

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.
For this implementation I used the features I extracted during the text pre-processing and feature extraction stage. I implemented this learning algorithm using the scikit learn module called sklearn.ensemble.AdaBoostClassifier. I set the number of estimators (The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.) to 500 and learning rate to 1.

The table below is a summary of the results on Bagging on Support Vector Machines:

Best Hyper-Parameter	Test Set Measurement Accuracy	Evaluation Set Accuracy- Kaggle
Learning rate γ: 1	Accuracy: 85% Precision: 85% Recall: 86% F1 Score: 85%	85.392%



5 Future Work

I will like to explore deep learning tools like Recurrent Neural Network and observe and compare it accuracy to the above learning algorithms.

References

[1] Sarkar, D. (2016). *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from Your Data* India: Apress Media.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).