CS6350- Machine Learning- Final Project

Hossein Moghimifam

Introduction

One of the most important applications of machine learning is Natural Language Processing. Here, we study the Large Movie Dataset V1.0[1]. The goal is to predict the correct label for each review so that we can divide them into good and bad reviews.

Different machine learning algorithms have been studied along with the data manipulation techniques. Perceptron showed a promising performance on the dataset. It is mostly because the speech is not i.i.d random variable. Improvements over the perceptron algorithm i.e. average perceptron and ensemble of average perceptron further improved the performance of the classifier. The ensemble of the average perceptron showed a better performance on the evaluation set than XGBoost.

Data preprocessing

Since the data is already extracted from the text and the bag of words model is already achieved, I will continue the work on the data from this point. No new feature would be added to the data. Checking the "Vocabs" file, we find out that the data is clean to some extent, there are no capitalized words in the features, there are no marks (.,!,?,/) in the features.

Four more steps can be done on the data to make it better. First, there is no point to have words such as that, this, the, a, etc. in the features. So, we would omit these features from the features. To do so, I use the 'stopwords' class form the nltk package. These words alongside other words that have occurred more than 20000 were inspected and the irrelevant ones were omitted: ['one', 'thi', 'hi', 'br', 'film', 'movi', 'wa']. This would cut the features from 74481 to 50889.

Second, there are words that are from the same root but are separate features right now. So I would also combine these to make a single feature. Words like absorb, absorbs, absorbed got all into a feature by summing up the number for each of them in a sentence.

Third, any word that has been occurred less than five times has been omitted. This would drastically reduce the number of features from 50889 to 18467.

Finally, since the repentance of the words in a text would impose a bias, we should solve this problem too. The longer the text the higher the feature count would be for it. To fix this issue, we can use Term Frequency (TF) instead of word counts and divide the number of occurrences by the sequence length. We can also downscale these frequencies so that words that occur all the time (e.g., topic-related or

[1] 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).

stop words) have lower values. This downscaling factor is called Inverse Document Frequency (IDF) and is equal to the logarithm of the inverse word document frequency[2].

$$\text{TF}(\text{word}, \text{text}) = \frac{\text{number of times the word occurs in the text}}{\text{number of words in the text}}$$

$$\text{IDF}(\text{word}) = \mathbf{log}\left[\frac{\text{number of texts}}{\text{number of texts where the word occurs}}\right]$$

$$\text{TF-IDF}(\text{word}, \text{text}) = \text{TF}(\text{word}, \text{text}) \times \text{IDF}(\text{word})$$

Algorithms

There are three sets of data available, train, test, and eval. I will use the test set as the development set and the eval set would be the actual test set. Any accuracies reported as the test accuracy are on the test set unless otherwise is mentioned.

Starting Point: Perceptron

Kaggle Submission: sub1.csv

One of the algorithms that provides a good accuracy on the raw data provided is perceptron. This is the only submission that has been done without changing anything on the data. So this would be a baseline for all the efforts that we would make on the data. It would result in accuracy of 0.865 for the test set and 0.72384 for the evaluation set.

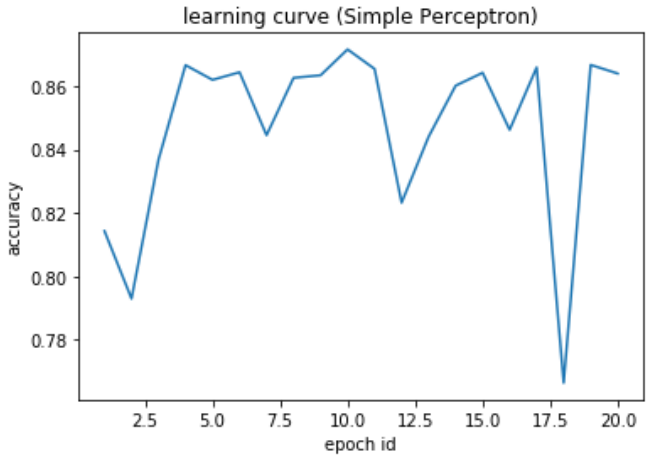
Cross Validation

Any cross-validation throughout the report is achieved by dividing the train set into 5 different sets and changing the hyper-parameters and finding the mean accuracy as the object of interest.

learning rate	total # of updates	mean accuracy	std
1	27409	0.832276	0.01228
0.1	27392	0.82738	0.01694
0.01	27496	0.847416	0.0097

Testing it on the test set using different epoch numbers would result in the learning curve below.

[2] https://medium.com/data-from-the-trenches/text-classification-the-first-step-toward-nlp-mastery-f5f95d525d73



Average Perceptron

Kaggle Submission: sub2.csv

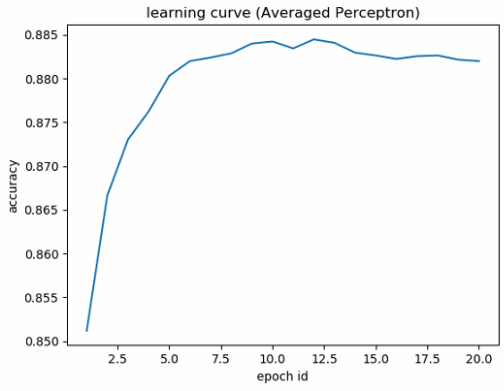
Another algorithm that provides a good approximation on the data is average perceptron.

Cross Validation

Any cross-validation throughout the report is achieved by dividing the train set into 5 different sets and changing the hyper-parameters and finding the mean accuracy as the object of interest.

learning rate	total # of updates	mean accuracy	std
1	27458	0.878408	0.003873
0.1	27367	0.879016	0.00309
0.01	27479	0.878168	0.00441

Testing it on the test set using different epoch numbers would result in the learning curve below.

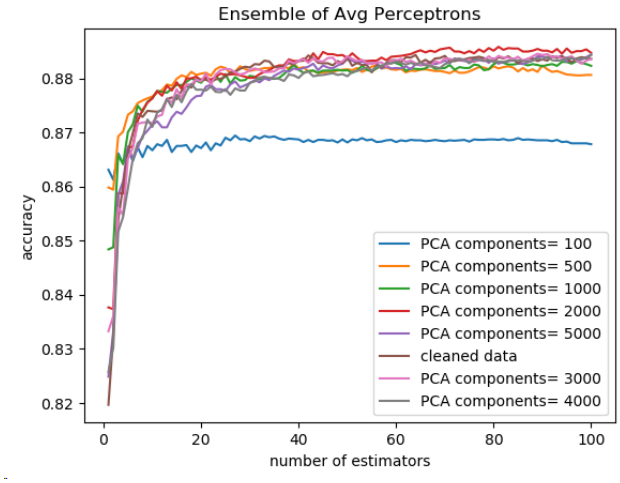


The highest accuracy on the test set is 0.88448. The accuracy of the classifier on the evaluation set would result in the accuracy: 0.8080. Using the same algorithm with the pre-processed data would result in the evaluation accuracy: 0.84992 (Sub3.csv). Furthermore, applying PCA with 100 components on the preprocessed data would result in the accuracy of 0.86240 (Sub4.csv).

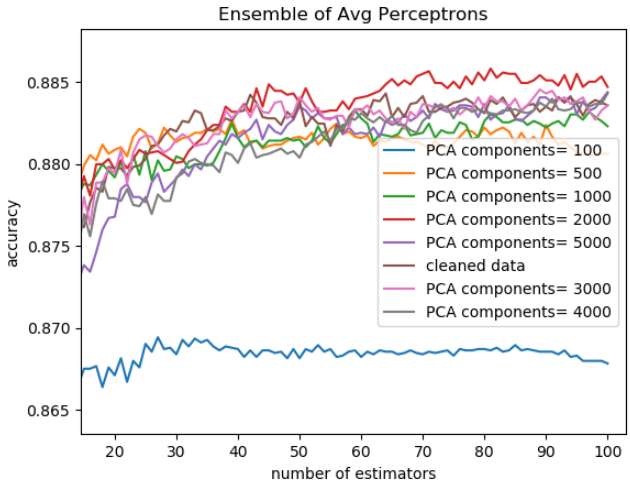
Ensemble of Average Perceptron

Kaggle Submission: sub11.csv

Using the voting prediction on the pre-processed data can be pretty robust and powerful. To do so, I will use numerous average perceptron classifiers to predict the label of the x. To train each average perceptron classifier, 20% of data is used each time. 5000 data points are selected randomly each time to train the classifier, then all the trained classifiers would vote on each data point with the same weight. The label would be chosen by the majority vote. To further improve the data, PCA is used to reduce the dimensionality of the feature space. Figures below show the accuracy of the test data with its respective data.



Closely looking at the accuracy in the upper band of the number of estimators would determine which data is best to be used for this classifier training. It shows that even though the results are pretty close, choosing PCA with 2000 number of components gives the highest accuracy. Actually, using 4800 components for the PCA would represent 84% of the variance in the data. Using 2000 components would represent 65% of the variance in the data.

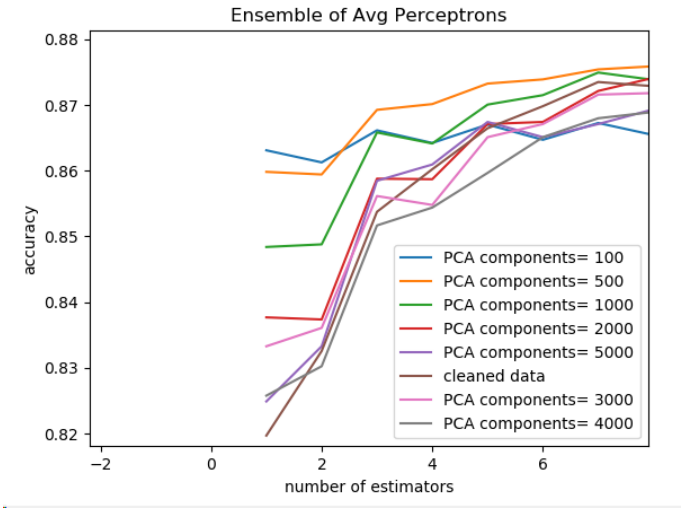


The ensemble of average perceptron classifiers with 100 number of estimators would predict the test data with the accuracy of 0.87760. This is the highest accuracy

SVM over Perceptron

Kaggle Submission: sub12.csv

Another way to improve the already good method of classification is using the 2-layer neural network. The predictions of the Perceptron would be used as the input to the SVM classifier. To do this within a computationally possible time, only one estimator of the Average Perceptron is used. Looking at the beginning of the PCA figure we would figure out that the PCA with 100 components would result in the highest accuracies. Using this data, the predictions for the test set is calculated using 200 average perceptron classifiers. The predictions would be used as the input for the SVM classifier.



Cross-Validation

Different values of hyper-parameters are tested to find the best combination for the classifier can be seen in the table below. Using the best hyper-parameters, the accuracy of the test set would be 0.87576. The accuracy of the evaluation set would be 0.86848.

learning rate	tradeoff	mean accuracy	standard deviation
0.0001	0.0001	0.87232	0.003297514
	.		
	.		
	.		
0.001	1	0.87256	0.003203498
0.001	10	0.87172	0.002344696
	.		
	.		
	.		
1	1	0.72236	0.295282438
1	10	0.5704	0.359786181

Logistic Regression

Kaggle Submission: Sub8.csv

Using logistic regression classifier is also another method to predict the labels for the dataset. The best hyper-parameters for the classifier are chosen using the 5-fold cross-validation. Using the logistic regression, the accuracies for the test and evaluation sets are 0.8392 and 0.81904 respectively.

learning rate	tradeoff	mean accuracy	standard deviation
0.0001	0.1	0.84328	0.003930598
0.0001	1	0.84332	0.003649877
	.		
	.		
	.		
1	100	0.84136	0.003418538
1	1000	0.84172	0.008515961
1	10000	0.85296	0.003810302

Checkpoint: XGBoost

Kaggle submission: sub10.csv

In the documentation for XG boost we can find: "**XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples."[3]

It seems to be an optimized method for classifying this problem. It can be a good comparison baseline for the classifiers that we've made. The cross-validation part is not shown, but the best hyper-parameters for the XGBoost classifier are 90 estimators with depth of 7. Using these hyper-parameters the XGBoost would predict the test set and the evaluation set with accuracies of 0.84216 and 0.86240 respectively.

Although it results in a high accuracy, our classifier of using ensemble of average perceptron classifiers would result in better predictions.

Further Improvement

Although we have shown a good performance on the data, we can further increase the accuracy. Feature extraction can be a big help in that regard. Combination of the words next to each other would be a good starting point. For example, using only single words as the feature would limit understanding of the text. The word "good" appears in both "good" and "not good". Combing these two together would increase the accuracy. Neural network can also be implemented to have a better prediction of the dataset.

[3] https://xgboost.readthedocs.io/en/latest/