Classifying Movie Reviews

Harshitha Manduva (U1080236)

February 13, 2019

PROJECT REPORT

Project description:
Each example in this classification task is a movie review. The goal is to predict whether the review is a positive or a negative one. The data used for this task is based on the Large Movie Review Dataset v1.0[1].

Algorithms implemented

I Implemented using 7 different algorithms and among which 5 of there are giving good accuracy on the eval data ranging above 85 percent but the Naive Bayes and the Id3 are not given a good performance on the data. Each of the algorithm is tried with multiple flavors and hyper parameters.The details about the implementations of each of the algorithm are provided in the respective part of this document.

- Simple preceptron
- Average perceptron
- SVM
- SVM with bagging
- Scikit learn nlp multi layer perceptron
- ID3
- Naive Bayes

Algorithms you used, their accuracy
Perceptron

I chose to implement 2 different variations of the perceptron mistake bound algorithm. The simple and the average perceptron implemented and each was run on multiple epocs.
The perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.
- Simple preceptron with margin = 1 and learning rate = 100

Train accuracy: **92.6520**
Max DEV accuracy: **87.2480**
Test or eval accuracy: this accuracy is obtained from kaggel **86.928**
Submission file name: **simplePerceptronOpFile.csv**
Submission name deception: **Final simple perceptron**

- Average perceptron with margin = 0 and learning rate = 100

Train accuracy: **92.8159**
Max DEV accuracy: **88.00**
Test or eval accuracy: this accuracy is obtained from kaggel **87.504**
Submission file name: **avgPerceptronOpFile.csv**
Submission name deception: **Final average perceptron**

SVM (support vector machines)

I implemented SVM in 2 ways. One is the simple SVM where the weight vector is created with all the elements and ran 20 epocs to determine the ultimate weight vector.
The other method is with the bagging of the SVM. I created 20 different SVM vectors by random sampling the training data by taking 1500 elements on each epoc and 20 epocs are run for each of the weight vector determination.

The Cross validation results are mentioned in the code commented out

Support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

- SVM with C = 10 and eta = 1

Train accuracy: **86.7120**
Max DEV accuracy: **84.6239**
Test or eval accuracy: this accuracy is obtained from kaggel **0.84736**
Submission file name: **svmOpFile.csv**
Submission name deception: **svm 1**

- Bagging in SVM with C = 10 and eta = 1

Train accuracy: **87.7720**
Max DEV accuracy: **84.9520**
Test or eval accuracy: this accuracy is obtained from kaggel **0.85440**
Submission file name: **svmBaggingOpFile.csv**
Submission name deception: **svm with bagging**

Scikit learn nlp multi layer perceptron

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset, where is the number of dimensions for input and is the number of dimensions for output. Given a set of features and a target , it can learn a non-linear function approximator for either classification or regression.

Scikit learn nlp multi layer perceptron

Train accuracy: **98.76**
Max DEV accuracy: **88.03**
Test or eval accuracy: this accuracy is obtained from kaggel **0.86272**
Submission file name: **nlpMultiLayer.csv**
Submission name deception: **multi layer nlp**

ID3

The ID3 algorithm is used by training on a data set S to produce a decision tree which is stored in memory. At runtime, this decision tree is used to classify new test cases (feature vectors) by traversing the decision tree using the features of the datum to arrive at a leaf node. The class of this terminal node is the class the test case is classified as.
- Id3 depth 11

Train accuracy: **65.83**
Max DEV accuracy: **61.7**
Test or eval accuracy: this accuracy is obtained from kaggel **0.54623**
Submission file name: **id2OpFile.csv**
Submission name deception: **Id3**

Naive Bayes

Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- Naive Bayes lambda 2

Train accuracy: **50.0**
Max DEV accuracy: **49.512**
Test or eval accuracy: this accuracy is obtained from kaggel **0.50127**
Submission file name: **naiveBayes.csv**

Submission name deception: **naive bayes**

Below are the details of the code implementation

For implementing the perceptron algorithm i used the Python3 programming language. And considering the data size Following are the design choices that I made. I used the array of dictionary to store the input data. where each row represent one independent data set. The dictionary is in the form of key as the feature name in this particular example(series of integers) and the corresponding value as the value of that dictionary element. And the variable data represents the set of these dictionaries in the form of an array. I added the expected label in the dictionary under the feature name 0. I made the choice to use the array of dictionary as it will be easy data retrieval and can Handel features effectively, i.e if the on e data row is missing on feature will not be effecting the other data row of the entire data set. This algorithm implementation cannot be done in arrays as the data is huge and that cannot be handled by RAM but the dictionaries will not have this issue. The weight vector is also maintained in the form of a dictionary. And the bias is added in this dictionary.

And for choosing the hyper parameters I segregated the data into different parts and created my own cross validation data. And used this data across the algorithms to find the respective hyper parameters for the algorithms.

What works, what doesnt work

For the perceptron implantation the margin and learning rate was chosen with trail and error. After deciding these 2 parameters the code was run on multiple epocs and choose the appropriate epoc when the dev accuracy was highest and ran the eval data on it and predicted the output.

I tried to remove the stop words from the feature vector as a method for reducing the features, and then implement the algorithms but that did not improve the performance of the algorithms so have reverted the changes.

I tried multiple hyper parameters for each of the algorithms and choose the best one with the the help of the cross-validation.

the Random forests is implemented but was taking too much time and no boost in the performance was given as the tree size that can be computed is not good enough for getting a good performance. And for this data considering the amount if features in the data set provided this algorithm did not perform well in the 20+ hours it executed.