CS590NLP Fall 2017 Homework 1

Mohit Gupta – gupta440@purdue.edu

TASK

The Problem statement is to write a program for sentiment analysis for the given data set. The task is to implement two models namely: Multi-Layer Perceptron (MLP or three-layer neural network) and Perceptron. The data set is given as ”sentences.txt” as the sentences and ”labels.txt” as the corresponding polarity labels, with positive label as ”1” and ”0” for negative label. The performance of the models are measured by F1 Score.

ALGORITHMS

(a) Perceptron

   (a) Create a dictionary of words by removing all characters except alphabets and space character from each line of the given dataset so as to clean the data.

   (b) Initialize the weight vector to be a V-dimensional vector where V is the size of the dictionary.

   (c) Create feature matrix (bag of words V-dimensional vector) for the given training sentences and feed it to perceptron. Bag of words features are used to avoid overfitting.

   (d) I use sigmoid of the dot product of the weight vector and feature vector as the output of perceptron

   (e) The prediction for a given sentence from the perceptron is 1 if sigmoid is greater than 0.5 and otherwise 0.

   (f) The loss function used is squared loss function and stochastic gradient descent is used to minimize this loss function.

(b) MLP

(a) For MLP, I used 9 nodes in the hidden layer with the activation function as sigmoid function with the loss function being squared loss function.

DataSet Distribution and Hyperparameter Tuning

(a) I divide the given dataset into 3 parts i.e. Training Dataset, Validation Dataset and Test Dataset.

(b) There were a total of 8530 sentences and my distribution was as follows:

   (a) Training Data - sentences from 1-2000 which had a positive label of 1 and sentences from 6501 - 8530 which had a negative label of 0.

   (b) Validation Data - sentences from 2001 - 3000 which had a positive label of 1 and sentences from 5501 - 6500 which had a negative label of 0.

   (c) Test Data - sentences from 3001 - 4265 which had a positive label of 1 and sentences from 4266 - 5500 which had negative label of 0.

(c) I used Validation Dataset to tune my model and get an understanding on how long I should train my models both Perceptron and MLP. I used the f-score from the unseen validation dataset after training my models on the training data.

(d) I tried both tf-idf weighting for my feature vectors and also min max normalization. Interestingly both on the validation dataset during training as well as the test dataset, tf-idf gives a slight improvement as compared to min max normalization. Note, that I have not removed stop words from the dataset and used tf-idf for the very same reason so as to give more importance to words which were unique to a sentence.

(e) I also randomized the selection of sentences during training since training on positive samples first and negative samples later was resulting in biasing resulting in poor training of the learning models.

(f) I tried 2 approaches to create a dictionary.

   (a) In the first approach, I created a dictionary of words from just the training dataset which meant that I had a lot of unseen words both for validation datasets during hyperparameter tuning as well for testing dataset.

   (b) In the second approach, I created a dictionary from the complete dataset including the blind dataset without the labels released on the day of submission. tf-idf worked better in this approach since, it helped in creating normalized feature vectors from test dataset.

(g) The value of learning rate for perceptron is 0.15 and I found this to be an optimal value when the model is trained in a loop which runs 18 times. In each loop, 1000 training examples are picked up randomly to train the model. The f-score observed on test dataset is between 0.71 to 0.72. This is so because I have tried not to overfit my model on the training dataset and have kept the squared loss value to be around 100 from the randomly picked 1000 sentences during training.

(h) For MLP, the value of both hidden layer learning rate and output layer learning rate is 0.5. The f-score for MLP on test dataset varies between 0.72 to 0.74. Similar to perceptron, I train the model in a loop which runs 20 times. In each loop, 4500 training examples are picked up randomly to train the model.

RESULTS on test dataset

Hyperparameters	f-Score Perceptron	f-Score MLP
learning rate 0.15	0.73	0.689
learning rate 0.30	0.68	0.69
3 learning rate 0.50	0.67	0.74

Thus, in the final submission, the learning rate for Perceptron has been kept at 0.15 and MLP has been kept at 0.50