CS57800: Statistical Machine Learning
HOMEWORK 2

Due: October 09, 2018 on Tuesday

1 Vanilla Perceptron

1.1 Algorithm

Algorithm 1 *Vanilla Perceptron Train Algorithm(D, NumEpoches, learningRate)*

```
w$_d = randomNumber(-1, 1)$ for all d=1...D
$bias = 1$
for epoch in range(NumEpoches) do
    shuffleData(D)
    for all (x,y) $\in$ D do
        $prediction = w_0x_0 + w_1x_1...w_Dx_D + b$
        if $prediction * y < 0$ then
            $w_d = w_d + y * LearningRate * x_d$, for all d = 1...D
            $b = b + LearningRate * y$
        end
    end
end
return $w_0, w_1...., w_D, b$
```

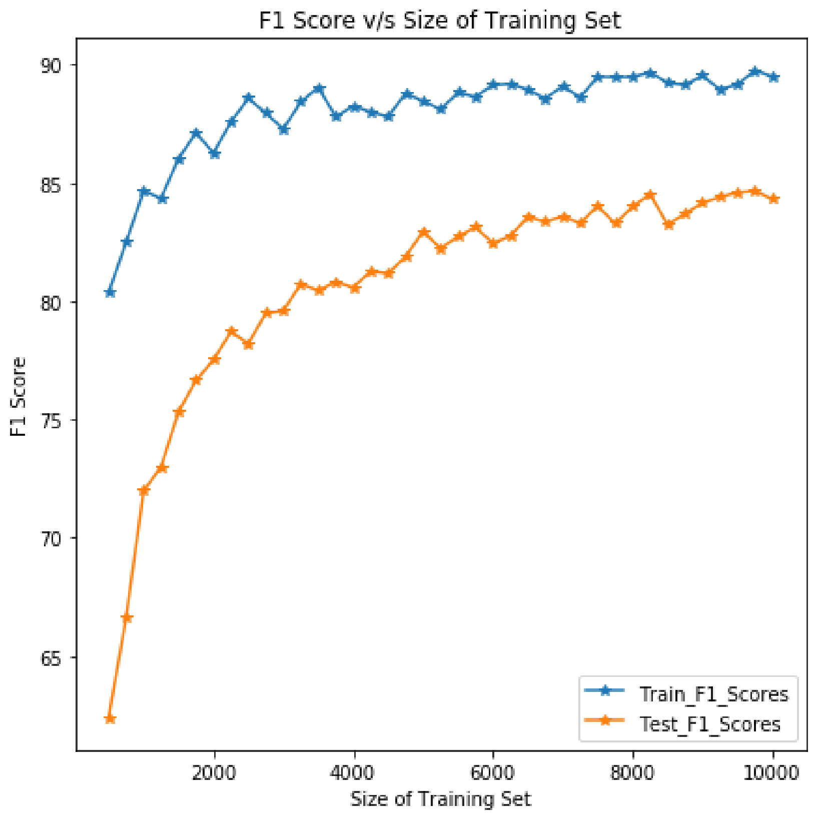
Algorithm 2 Inference Algorithm(w$_{00}, w_{01}..w_{0D}, w_{10}..w_{1D}...w_{90}..w_{9D}, b_0..b_9x$)

```
For all 10 perceptrons/winnows, calculate
a$_p = w_{p0}x_0 + w_{p1}x_1... + w_{pD}x_D + b_p$
return $argmax(a_0, a_1...a_9)$
```

1.2 Implement

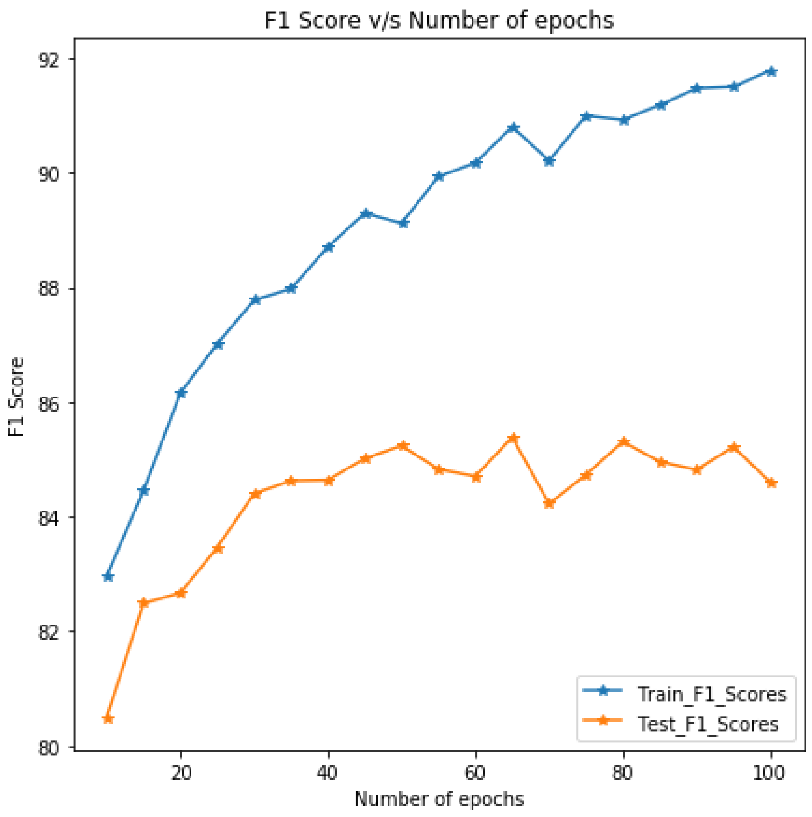
The implementation can be found inside the file vanilla_perceptron.py. Kindly note that I have util functions which are in util.py mainly having functions used for preprocessing, performing the inference step and calculating the f1 score. The actual logic of calculating f1 score is in f1_macro.py.

1.3 Effect of size of training set on learning Vanilla Perceptron



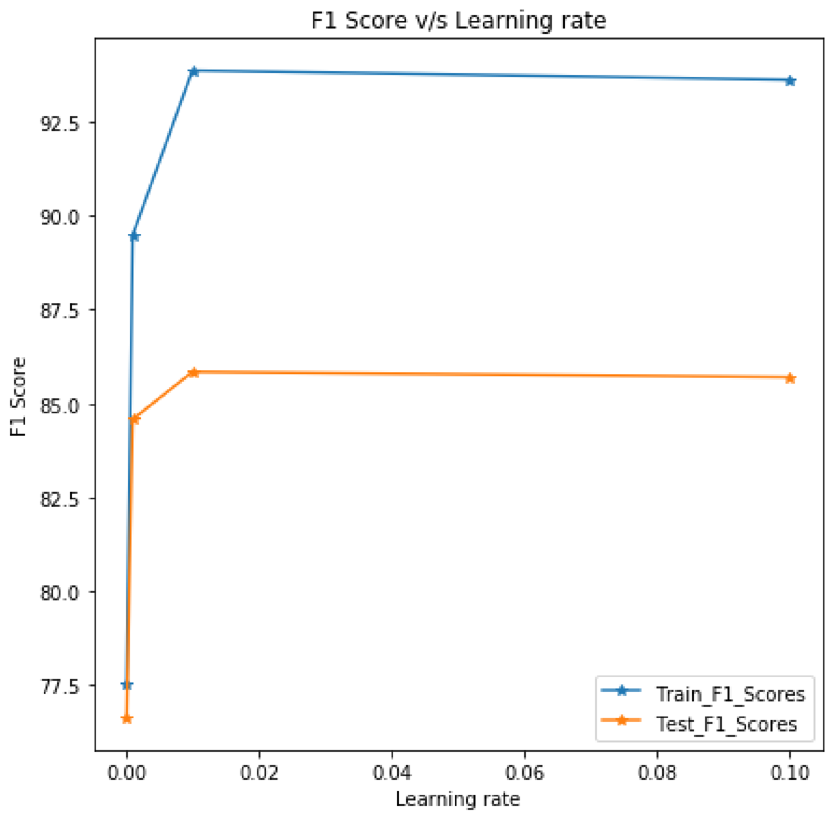
As we can see, the scores improve with the increase in the training set and then become flat. The best scores on the test set are around 84% f1 score for training set size of more than 8000.

1.4 Effect of the number of epoch in learning



We can observe that the performance increases consistently till 50 epoches. After that, the model starts to overfit and there is either a decrease or flattening of performance on the test set even though the f1 score on training set keeps on increasing. The best scroes are around 85% f1 score for 50 epochs.

1.5 Effect of learning rate for the values .0001, .001, .01, .1



We can observe that the best performance numbers are observed for learning rate 0.01 which looks to be the optimal learning rate for perceptron to learn on the training data set and gives f1 score of around 85.5% on test set.

1.6 Report your observations on the learning curves

After tuning the hyperparameters with the above exercise, we observe that the best results are around 85.5% f1 score for vanilla perceptron algorithm for hyperparameters learning rate = 0.01, epochs = 50 and training data size = 10000.

2 Average Perceptron

2.1 Average Perceptron algorithm

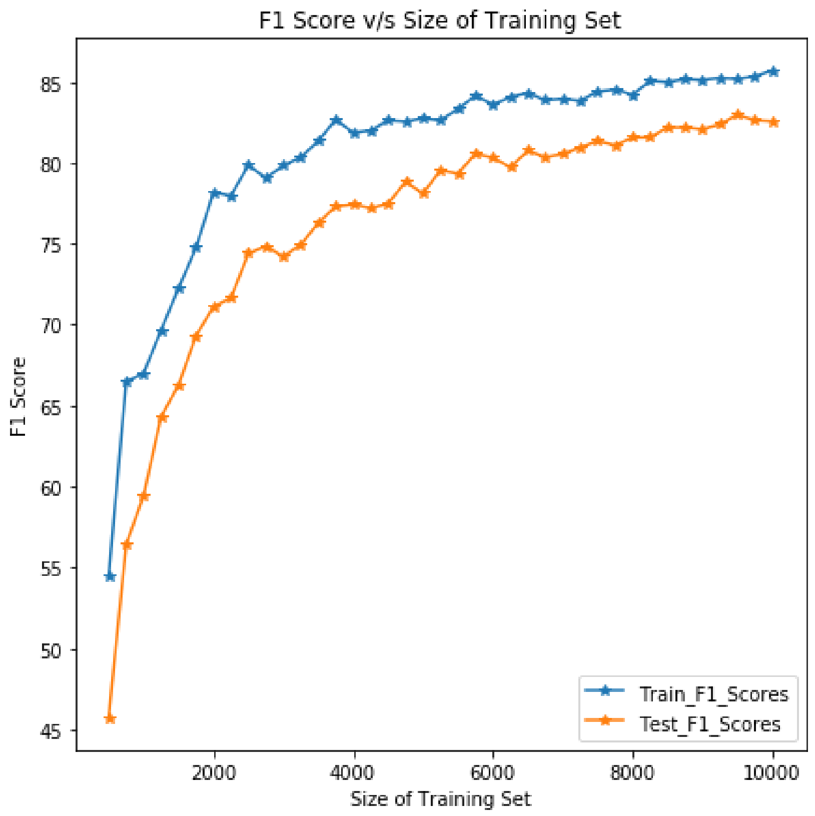
Algorithm 3 *Average Perceptron Train Algorithm(D, NumEpoches, learningRate)*

```
w_d = $randomNumber(-1, 1)$ for all d=1...D
$avgWeights_d = 0$ for all d=1...D
$bias = 1$
for epoch in range(NumEpoches) do
    shuffleData(D)
    for all (x,y) ∈ D do
        $prediction = w_0x_0 + w_1x_1...w_Dx_D + b$
        if prediction * y < 0 then
            $w_d = w_d + y * LearningRate * x_d$, for all d = 1...D
            $b = b + LearningRate * y$
            end
            $avgWeights_d = avgWeights_d + w_d$ for all d = 1...D
            end
        end
        return $avgWeights_0, avgWeights_1...., avgWeights_D, b$
```

2.2 Implement

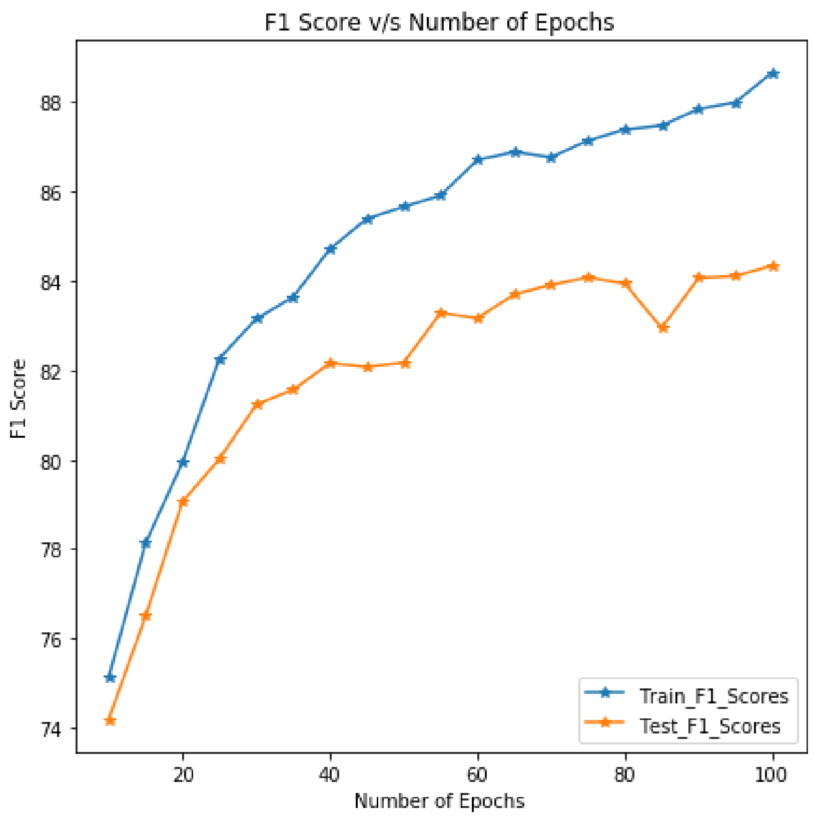
The implementation can be found inside the file average_perceptron.py. Kindly note that I have util functions which are in util.py mainly having functions used for preprocessing, performing the inference step and calculating the f1 score. The actual logic of calculating f1 score is in f1_macro.py.

2.3 Effect of the size of training set in learning Average Perceptron



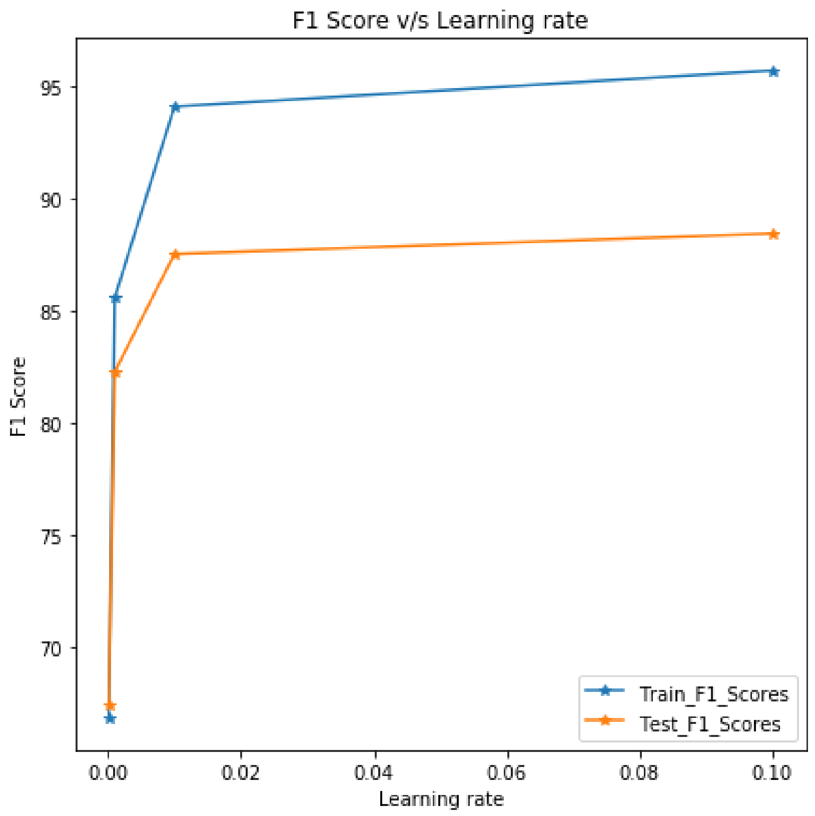
As we can observe, the average perceptron algorithm learns with increase in size of training set, with best f1 test scores at around 83% after training data set size becomes greater than 9000. One thing to observe is that the learning rate 0.001 could be less for average perceptron since the scores didnt flatten which means that it might be underfitted. We can confirm this after training with higher learning rates for example 0.1 and see the results.

2.4 Effect of the number of epoch in learning



We observe that the best scores are at around 100 epochs i.e. f1 score of around 84.3%. One observation could be that average perceptron with learning rate 0.001 needs high number of epochs to train since even after a very high number of epochs i.e. greater than 90, scores are still increasing for test set and are not flattening.

2.5 Effect of learning rate for the values .0001, .001, .01, .1 in learning



Here, we observe that average perceptron gives the best results for learning rate 0.10 i.e. around 88% f1 score.

2.6 observations on the learning curves

After playing with the hyperparameters as given above, we get the best scores for average perceptron of around 88.6% f1 score with training size = 10000, epochs = 50 and learning rate = 0.1

3 Winnow

3.1 Winnow algorithm

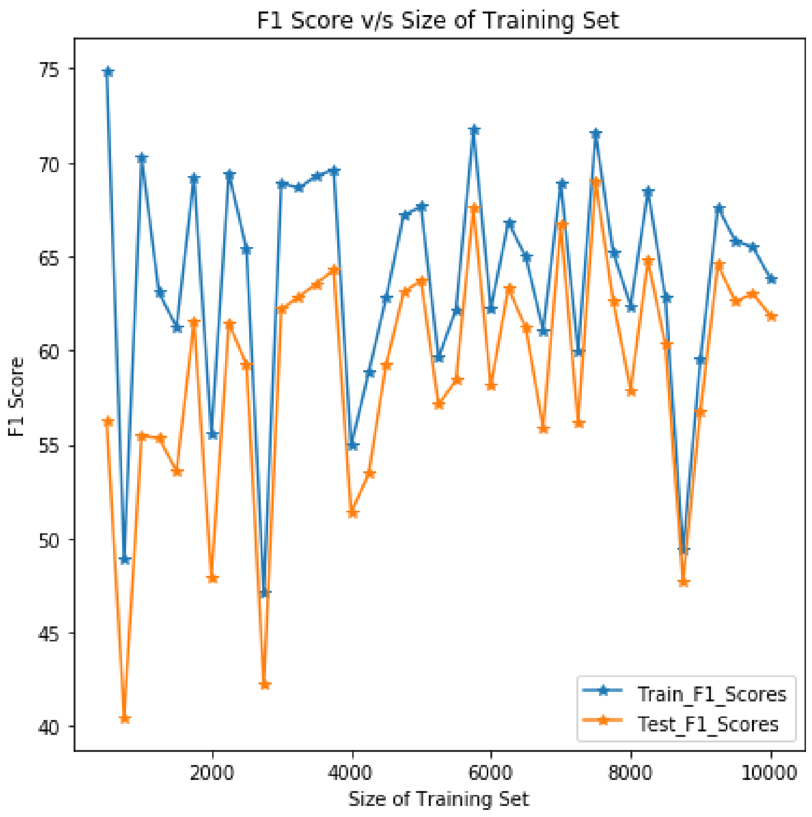
Algorithm 4 *Winnow Train Algorithm(D, NumEpoches, factor, threshold)*

```
w_d = $randomNumber(0.1, 1)$ for all d=1...D
$bias = 1$
for epoch in range(NumEpoches) do
    shuffleData(D)
    for all (x,y) $\in$ D do
        prediction = $w_0 x_0 + w_1 x_1 ... w_D x_D + b$
        if y = 1 and prediction < threshold then
            $w_d = w_d * factor,\ for\ those\ d\ in\ 1, 2, 3..D\ where\ x_d\ =\ 1$
            $b = b * factor$
            end
            else if y = -1 and prediction >= threshold then
                $w_d = w_d / factor,\ for\ those\ d\ in\ 1, 2, 3..D\ where\ x_d\ =\ 1$
                $b = b / factor$
                end
            end
        end
        return $w_0, w_1 ...., w_D, b$
```

3.2 Implement

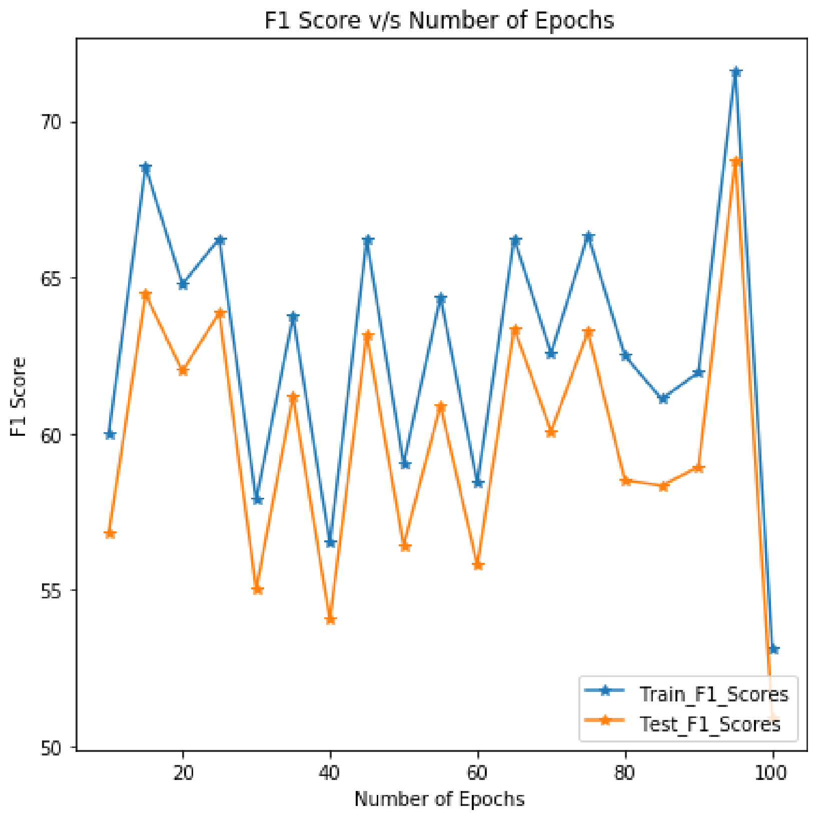
The implementation can be found inside the file winnow.py. Kindly note that I have util functions which are in util.py mainly having functions used for preprocessing, performing the inference step and calculating the f1 score. The actual logic of calculating f1 score is in f1_macro.py.

3.3 Effect of the size of training set in learning Winnow



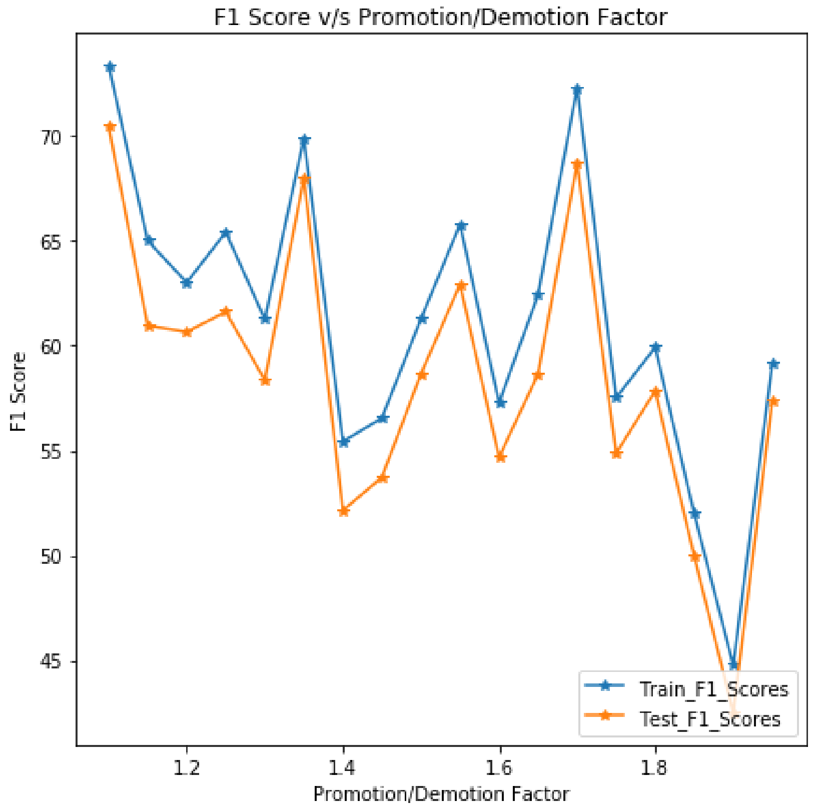
The default values were epochs = 50, factor = 1.55, threshold = 785

3.4 Effect of the number of epoch in learning



The default values were training size = 10000, factor = 1.55, threshold = 785

3.5 Effect of change in factor in learning



The default values were training size = 10000, factor = 1.55, threshold = 785

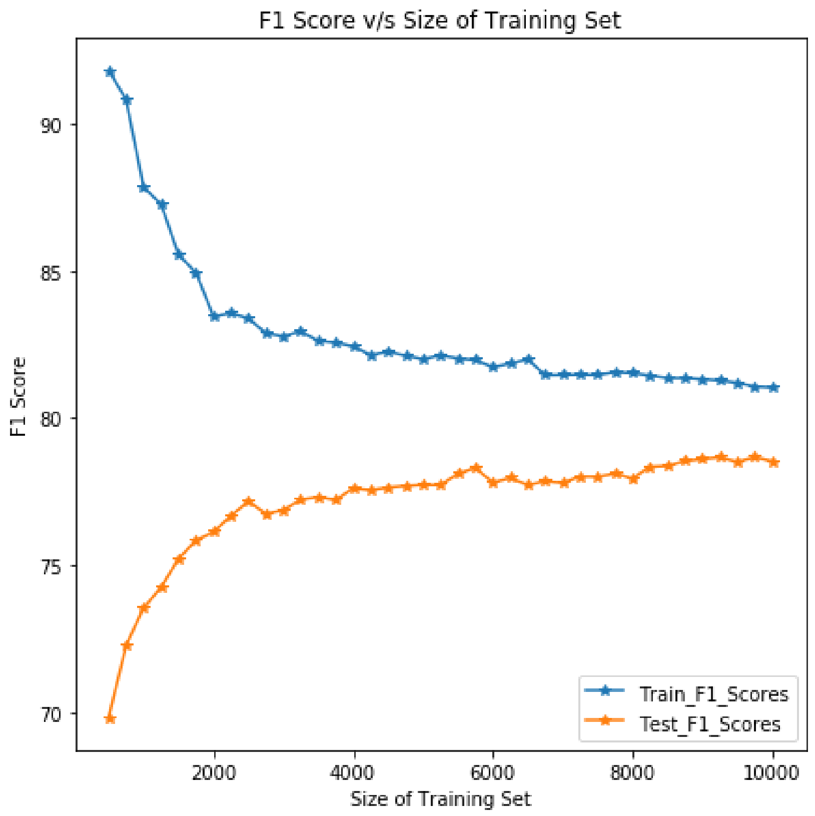
3.6 observations on the learning curves

The winnow algorithm is giving test set f1 scores of around 50-70% with the best scores of around 70% f1 scores got for hyperparameters - promotion/demotion factor being 1.1, epochs = 50 and training data size = 10000 and the threshold being 785. That said, we observe that the results for winnow algorithm are unstable in comparison to perceptron algorithm. For a particular set of hyperparameters, they vary in the range of 55-70% f1 score.

On the other hand, the results of average winnow algorithm seem much better. The algorithm is similar extension as from perceptron to average perceptron.

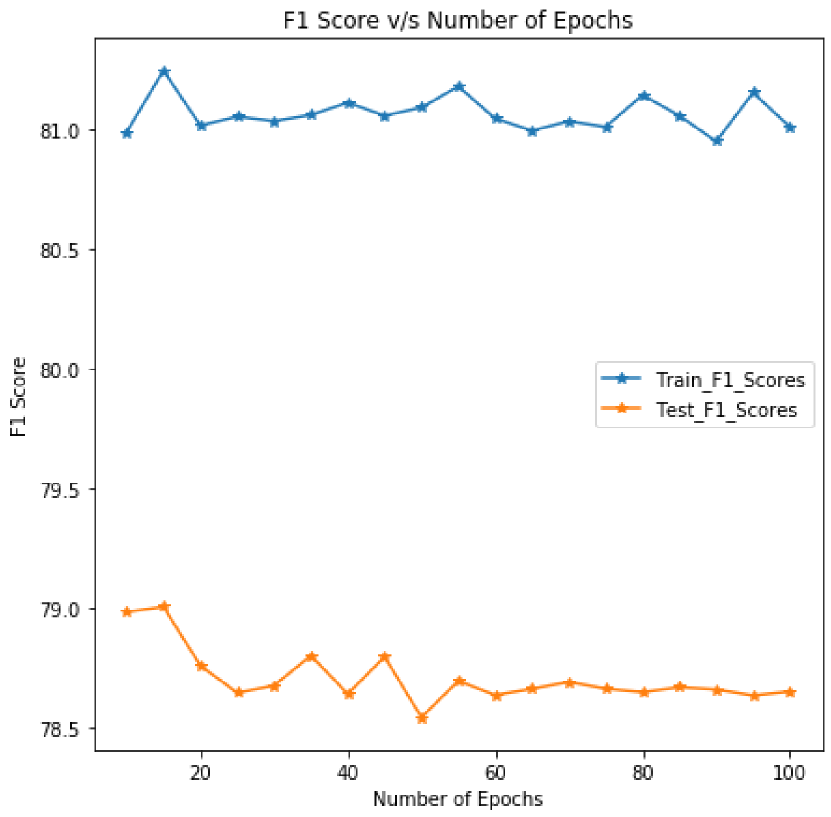
The best results for average winnow algorithm are around 79% f1 score on test data set with hyperparameters - factor = 1.55, size of training set = 10000, epochs = 50 and threshold = 785. The learning curves for average winnow algorithm are given below:

3.7 Effect of the size of training set in learning Average Winnow



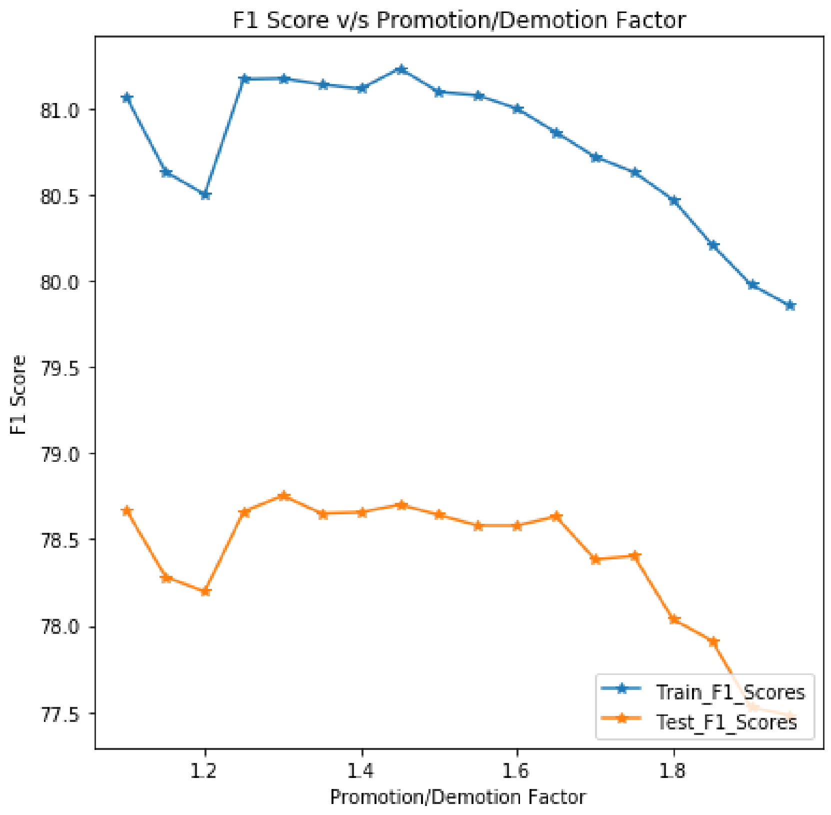
The default values were epochs = 50, factor = 1.55, threshold = 785

3.8 Effect of the number of epoch in learning



The default values were training size = 10000, factor = 1.55, threshold = 785

3.9 Effect of change in factor in learning



The default values were epochs = 50, training size = 10000, threshold = 785

4 Open Ended Questions

4.1

Based on the learning curves, we observe that Average Perceptron algorithm performs the best out of vanilla perceptron, average percetron and winnow algorithm. The best scores for average perceptron algorithm are around 88.6% f1 score on the test data set. The best performance for vanilla perceptron are around 85.5% f1 score on test set and winnow's best scores around 70% f1 scores on test set with average winnow giving 79% f1 scores on test set.

4.2

On hypothesis class C and with n features, we call an algorithm mistake bound if the no of mistakes which the algorithm makes are polynomial in the size of n i.e. our feature space.

4.3

For learning Boolean conjunctions, a simple mistake bound algorithm is to initially have all the literals (say) x_1, x_2 ... x_n as the candidates.

Now the only difference between algorithm for boolean conjunctions and that for monotone conjunctions as discussed in slides is the following:
In boolean conjunctions, we can have not of x_i i.e. $-x_i$ also in the formula. So, the only difference in this algorithm is that whenever we get a training example which has output 1 but say the literal x_i is not active in that example, we do not eliminate this literal but replace it with not of x_i i.e. $-x_i$. Doing this will make our candidate set consistent with the training example. We thus proceed and read all training examples by following the above mentioned replacement procedure.

This algorithm is mistake bound since the max number of replacements that we can make is equal to the number of literals i.e. the size of our feature space n. This property is sufficient proof of our algorithm being a mistake bound algorithm.

4.4

Yes, since the data is linearly separable, the perceptron algorithm will converge in both the situations. There is advantage of selecting examples randomly only when the data is not linearly separable which is not the case here.

The training errors for both the classifiers will be 0. The perceptron will be able to divide the data using a hyperplane as it is linearly separable which translates to 0 training error.