Report

1)

Single Simple perceptron without margin:

we initialize the weight vectors to 0 and update each time a wrong classification occurs.

In this case the data takes **13** epochs for converging, and it predicts with **1.0** precision and **1.0** recall values.

Single Simple perceptron with margin:

In this, we change the checking condition from <=0 to <=b*modulus(w),

where b is the bias value and w is the weight vector.

It is found that in this case the data takes **13** epochs for converging, and it predicts with **1.0** precision and **1.0** recall values.

It is seen that having a small bias has no significant effect in this case. But if the bias value is increased the precision and recall sway a bit from a simple perceptron.

Single Batch perceptron without margin:

In this, we store all the mis-classifications in a single epoch and update at a time after the iteration is done.

It is found that in this case the data takes **2184** epochs for converging, and it predicts with **1.0** precision and **1.0** recall values.

In this case we see that a batch perceptron is performing way bad in terms of number of epochs.

Single Batch perceptron without margin:

In this, we store all the mis-classifications in a single epoch and update at a time after the iteration is done in addition to the conditioan.

It is found that in this case the data takes **2185** epochs for converging, and it predicts with **1.0** precision and **1.0** recall values.

In this case we see that a batch perceptron is performing way bad in terms of number of

epochs.

Question 2

Relaxation + Margin:

Here we assign weight to each of the words and set a threshold on the number of epochs and update the weight vectors till those many epochs.

We see that on the test data the hyperplane learnt in this manner has a precision **0.1** and a recall value of **0.75**.

Modified :

This is a simple incremental perceptron, exception being that each time we update the weight vector, we store the number of cases where the previous weight vector has

We see that on the test data the hyperplane learnt in this manner also has a precision **0.1** and a recall value of **0.75**.

Question 3

Algorithmic Implementation

Here, we collect all the data and find the entropy based on each classification and classify the data based on that test and then classify the classified data again till we reach the limit.

For the sample test case given, the accuracy is 0.9 and for prediction data the accuracy is 0.75

Question 4

Algorithmic Implementation

The files are iterated once and a count of the unique words is got. Then we make a feature vector for each word and we have a feature indicating the frequency of each

word. We then find the euclidean distance and output the class using knn.

Results

The accuracy for various k values.

For k = 1, accuracy is **0.7** I.e 7/10 are classified correctly.

For k = 5, accuracy is 0.6

For k=40 – 100 accuracy drops down to **0.8.**