CS 550- HW2- Multilayer Perceptrons

Oytun Gunes, ID: 20901170

November 17, 2017

In this homework, I have used multilayer perceptrons, which is an effective method for the classification problem. I am going to investigate how adding layer to the network affects the performance, and I will optimize my parameters in order to increase the accuracy. To define our problem we are using linear discriminants for the classification, and what we would like to do is to contruct a linear function from the inputs (feature vector) x to the targets (labels) y as:

$$f(x) = \hat{y} = w^T x + w_0 \tag{1}$$

Where w is the weigth vector. We are defining a loss function, and our aim is to find weight vector that minimizes the loss function. This problem does not always have analytical solution. Therefore we need an iterative optimization method. Gradient descent is a commonly used optimization method. It is basically as follows:
1.start with random weight vector
2.do

$$\Delta W_i = -\eta \frac{\partial loss_A LL}{\partial W_i} \qquad forall \;\; i \tag{2}$$

$$W_i = W_i + \Delta W_i \qquad forall \;\; i \tag{3}$$

3.until it converges, where is the learning rate.

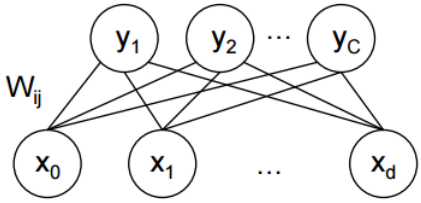

Figure 1: Muti-class classification in neural networks

1 Part 1a)

1- My loss function is sum of squares error as:

$$loss = \frac{1}{2} \sum_k (\hat{y}_k - y_k)^2 \tag{4}$$

2- The update rule when the stochastic gradient is used is:

$$\Delta W_i = \eta \sum_t (y^t - \sigma(net^t))\sigma(net^t)(1 - \sigma(net^t))x_i^t \tag{5}$$

2 Part 1b)

1- Learning rate: $\eta = 0.066$
2- Max validation failures: $f = 2$

I have used %80 of the training data to train the neural network, and %20 of the data in order to validate the performance and optimize the parameters. Once I fixed the number of hidden units, and one parameter, I

performed grid search in order to find the optimum value for the remaining parameter. Number of maximum validation failures means that in the last x epochs validation accuracy remained almost constant. The parameters vs. validation accuracies are as follows:

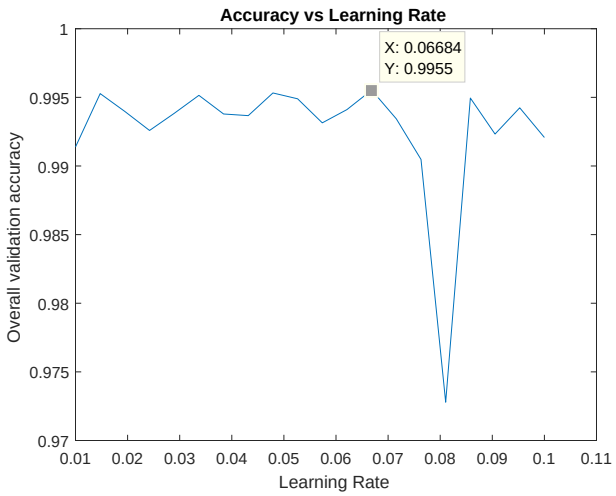

Figure 2: Accuracy vs Learning Rate

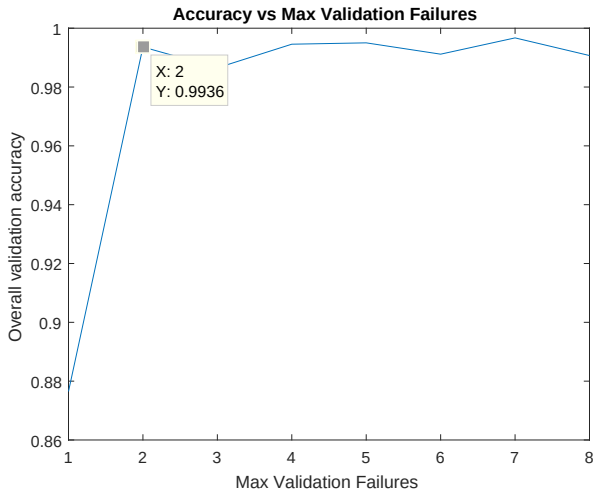

Figure 3: Accuracy vs Max Validation Failures

I have tried to find a high learning rate which gives a higher accuracy, and a lower max validation failures with respect to validation accuracy in order to speed up the algorithm.

3 Part 1c)

The initial data was unbalanced. In the class we have learnt that neural networks may have difficulties in learning the minority class and they may favor the majority class. In order to deal with this problem and make the data balanced, there are two ways: oversampling and undersampling. I have preferred oversampling the minority classes in order not to lose data.

Table 1: Oversampling on the unbalanced data

	Number of Instances		
	Class 1	Class 2	Class 3
Before balancing the data	93	191	3488
After balancing the data	2883	3056	3488

Scaling and normalizing the feature was another issue in practice in order to prevent the classifier to prefer some features over the others. In the class we have learnt that a neural network adjusts weights in favor of features with smaller magnitudes. Therefore for all features x_i I have applied:

$$normalized x_i = \frac{x_i - \mu_i}{\sigma_i} \tag{6}$$

4 Part 1d)

I have conducted many experiments to examine the accuracies for training and test data. The tables are below. I have tried 7 different hidden unit numbers to obtain train and test accuracy. It seems that 5 is the optimum

Table 2: Accuracies on training data

	Training Accuracies			
Hidden Unit Number	Class 1	Class 2	Class 3	Overall
4	1	1	0.98487	0.99439
5	1	1	0.98516	0.99449
6	1	1	0.98344	0.99386
7	1	1	0.98401	0.99407
8	1	1	0.98202	0.99333
9	1	1	0.97602	0.9911
10	1	1	0.97745	0.99163

Table 3: Accuracies on test data

	Test Accuracies			
Hidden Unit Number	Class 1	Class 2	Class 3	Overall
4	0.80822	0.89831	0.98836	0.97987
5	0.75342	0.53107	0.99371	0.9647
6	0.78082	0.89266	0.99308	0.98337
7	0.73973	0.87006	0.99465	0.98279
8	0.68493	0.52542	0.99559	0.9647
9	0.72603	0.40678	0.99528	0.95916
10	0.64384	0.42373	0.99622	0.95916

number for the training data, and 6 is the optimum number for the test data. The test accuracy increases as hidden number increases up to 6, after 6 it decreases.

5 Part 1e)

5.1 Normalization

I have normalized the training data and obtained accuracies and did not normalized the trainingd ata and obtained test accuracies as follows. In this case, as expected normalizing training data gives a higher test accuracy.(Table 4) |

Table 4: The effect of normalization on the data

	Test Accuracies			
	Class 1	Class 2	Class 3	Overall
Normalized	1	0.99476	0.98516	0.9928
Not Normalized	1	0.99476	0.98401	0.99237

5.2 Balancing the data

I have performed oversampling in order to make the class distribution balanced. After making it balanced I have obtain the results as in Table 5.

Table 5: Oversampling on unbalanced data

	Number of Instances		
	Class 1	Class 2	Class 3
Before balancing the data	93	191	3488
After balancing the data	2883	3056	3488

Table 6: The effect of making the data balanced

	Test Accuracies			
	Class 1	Class 2	Class 3	Overall
Balanced	1	1	0.9863	0.99492
Unbalanced	0.7957	0.44503	0.99599	0.96315

I was expecting the test accuracy of balanced data is higher, it can be seen from Table 6 it is as expected.

5.3 Stochastic-Batch-Mini-Batch

In stochastic learning the weight update is performed over each instance. For the batch learning it uses all of the instances to update the weights, which is mostly unnecessary and loss of time and computation for the algorithm. In this case, mini-batch is a good tradeoff which selects some of the instances randomly to update weights.

6 Part 2)

6.1 Update Rule and Parameters

There is no change in my update rule and the parameters, it is the same with part1.

$$\Delta W_i = \eta \sum_t (y^t - \sigma(net^t))\sigma(net^t)(1 - \sigma(net^t))x_i^t \tag{7}$$

Parameters:
1- Learning rate: $\eta = 0.066$
2- Max validation failures: $f = 2$

Table 7: Accuracies on training data when two layers are used

Hidden Unit Number		Training Accuracies			
Layer1	Layer 2	Class 1	Class 2	Class 3	Overall
4	4	1	0.98953	0.98116	0.98962
4	5	1	1	0.9823	0.99343
4	6	1	1	0.99058	0.9965
5	4	1	1	0.99543	0.99831
5	5	1	1	0.98944	0.99608
5	6	1	1	0.99086	0.99661
6	4	1	1	0.99515	0.9982
6	5	1	1	0.99372	0.99767
6	6	1	1	0.98287	0.99365

Table 8: Accuracies on test data when two layers are used

Hidden Unit Number		Test Accuracies			
Layer1	Layer 2	Class 1	Class 2	Class 3	Overall
4	4	0.90411	0.94915	0.99465	0.99037
4	5	0.84932	0.9435	0.99497	0.98921
4	6	0.63014	0.44633	0.99434	0.95828
5	4	0.76712	0.67797	0.99308	0.972
5	5	0.67123	0.84181	0.9915	0.97695
5	6	0.86301	0.88701	0.99591	0.98746
6	4	0.84932	0.9096	0.99245	0.98512
6	5	0.93151	0.9548	0.99654	0.993
6	6	0.64384	0.48023	0.99339	0.95945

7 Part 3)

To comment on the tables above, if we compare the multi-layer case for the training data, for the 4 hidden unit case with one hidden layer the training accuracy was 0.99439 and with two hidden layer with same hidden units it is decreased to 0.98962, which is expected since it has started to overfit data.

However, for the test data single hidden layer case with 4 hidden units, the accuracy was 0.9787 and it increased to 0.99037. I was expecting an increase slightly since in this way the weight matrices fits the data better in a deeper way.