Intro2MachineLearningHW03

The code I used for importing the training data is the same as the past homework.

X: The training data set (125*320)

x_test: The data set which will be tested after the training. Input layer. (70*320)

Y_truth: label set of training data which we manipulated for multiple class. Output layer. (125*5)

y_truth_test: label set of test data. (70*1)

K: class number (5)

N: training data sample number. (125)

D: dimension number in training data set. (320)

H: Hidden node number without the bias. (20)

I got sigmoid, softmax and safelog functions from the previous lab sessions. Then I changed the loss, gradient functions in order to satisfy the backpropagation algorithm.

Z: The nodes in the hidden layer (125*20)

W: The weighted edges connecting from the input (X) to the hidden layer (Z). (321 * 20)

V: The weighted edges connecting from the hidden layer (Z) to the output layer (Y_truth). (21*5)

I defined the loss function as the negiative sum of the multiplication of Y_truth and safe_log of y_predicted. I defined gradient of v in a for loop from k=1 to 20, that iterates over the rows of V (1*5). In order to find a vector with these dimensions, I found the difference in the truth and the predicted value as a matrix of 125*5 then I replicated Z[,k] which is a 125*1 matrix to be 125*5 and multiplied them element wise. After that I summed the columns which lead me to the 1*5 vector. At the end I used the sum and the multiplication in the book.

I defined gradient of W in a for loop from h=1 to 20, that iterates over the columns of W (320*1). In order to find a vector with these dimensions, I found the difference in the truth and the predicted value as a matrix of 125*5, multiplied it element wise with a matrix I replicated from 1*5 to 125*5 by column, which was the v[h+1,](1*5). The plus 1 is because the first value is from the bias which will be always 1. After The multiplication I had a matrix of 125*5. I row summed it to find a 125*1 vector. Then I multiplied this vector element-wise with Z[,h] and (1 – Z[,h]) which are both 125*1. After the multiplication, I replicated this vector to be a 125*321 matrix by column. Then I multiplied this matrix with cbind(1, X) (125*321)(1^{st} column is the bias term) and acquired a matrix of 125*321. Then I column summed it to find the vector of 1*321. At the end I used those 2 sums and the element wise multiplications in the book.

W and V values generated randomly at first as uniform values. Then, we use gradient descent to find the minimum error value or the maximum iteration number. After the loss function is satisfied or the maximum iteration is achieved, we now completed training our data. After getting the confusion matrix for the training data itself the I label the test data with the W and V weights I found. The output is the following:

```
> print(confusion_matrix)
            y_truth
Y_predicted  1  2  3  4  5
          1 25  0  0  0  0
          2  0 25  0  0  0
          3  0  0 25  0  0
          4  0  0  0 25  0
          5  0  0  0  0 25
> print(confusion_matrix1)
                 y_truth_test
Y_predicted_test  1  2  3  4  5
               1 13  1  0  0  0
               2  1 13  0  0  1
               3  0  0 14  0  1
               4  0  0  0 14  0
               5  0  0  0  0 12
```