

## **Machine Learning Project 3: CLASSIFICATION**

UBitname: manishat

Person number: 50207628

### Introduction:

As a part of this project, it was asked to implement and evaluate classification Algorithms. The various classification algorithms I worked on as a part of this project are:

1. Logistic Regression
2. Single Layer Neural Network
3. Convolutional Neural Network

### Datasets provided:

The datasets provided to work on were MNIST dataset and USPS dataset.

For MNIST Dataset, I divided the datasets into training, validation and test data sets. The corresponding datasets(train, valid and test) are the zeroth and (target) first elements of the array training\_data, test\_data and validation\_data are divided into.

### Logistic Regression:

For Logistic regression, the procedure I followed was to generate aW vector with random weights generation. The bias (B) matrix I used was a unit matrix with the corresponding sizes.

I compute A matrix by the formula  $A = A + Bk$ , the initial value of A is calculated by using  $A = \text{np.dot}(\text{train}, w0)$

Then the compute the Yk matrix and retrieve the target matrix from the corresponding dataset.

I then calculate the Delta W matrix by calculating the dot product of transpose of zeroth matrices and the difference matrix.

The difference matrix is calculated from Yk and modified new matrix.

I arrived at the value of eta to be 0.4 in order to get better results after a lot of trial and error calculations.

I now calculate the error value and then correspondingly add it to the weights and see the number of iterations for which value between initial and final error is reduced to a minimum.

Then I calculate the correctness using the number of matches between new Yk vector and target vector and calculate the percentage of correctness.

For logistic regression, the result I got was 85% when it ran for 100 iterations. Similar results have been obtained when the input given was validation data and testing data.

This is how the output screen looks like:

```
MATRIX A
[[ 50.65393695  56.91061117  55.09852733 ...,  56.59213293  54.31712876
  59.42986655]
 [ 58.55573826  64.0382199   62.32532645 ...,  63.38538059  63.3176243
  68.00990328]
 [ 34.82561139  36.19684506  39.4059578   ...,  37.90673281  34.15884092
  43.10461842]
 ...,
 [ 46.40269808  46.27064424  47.42233626 ...,  44.26313798  41.55444123
  47.27399748]
 [ 42.76667091  46.94320588  48.49631738 ...,  47.69224046  43.62167411
  45.7200828 ]
 [ 47.74432959  58.89488148  57.76515577 ...,  55.2712174   49.73082538
  54.08038702]]

TT
new
[[ 0.  0.  0. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  1.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  1.  0.]]
(50000, 10)
DWE
[[ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]]
[[ 0.36414364  0.23717154  0.10158213 ...,  0.15537311  0.21019479
  0.95555261]
 [ 0.77149701  0.68146728  0.46872454 ...,  0.46561882  0.78369832
  0.64354495]
 [ 0.56270136  0.09248167  0.82833825 ...,  0.77881743  0.5790207
  0.92910298]
 ...,
 [ 0.3573971   0.50907902  0.56575192 ...,  0.07649966  0.79298627
  0.64867884]
 [ 0.01947029  0.06730449  0.87329273 ...,  0.02112867  0.45216464
  0.22107689]
 [ 0.8073422   0.21172523  0.84589044 ...,  0.39941444  0.04809704
  0.78395395]]
[ 3.  0.  4. ...,  8.  4.  8.]
89
```

### **Single Neural Network:**

For the single neural network, theoretically, the more number of hidden layers, the better is the performance of the classification algorithm. Here each column of the training dataset is being taken as the input for the single neural network and a single hidden layer is being used. Hence the number of nodes in the order are 784, any arbitrary number(M) and ten nodes respectively.

I tried to compute  $Z_j$  using  $X, W$  and  $B$ . I use softmax function to calculate the  $Z_j$  value which is of size  $1 \times M$ .

Then I compute the  $A_k$  matrix using  $X_j, W_{kj}$  and  $B_k$  using the formula  $A_k = Z_j * W_{kj} + B_k$

Later, I compute the  $Y_k$  value using softmax of the elements in  $A_k$  matrix.

Computing the delta matrix can be done by subtracting the  $Y_k$  from  $T_k$  (Target matrix)

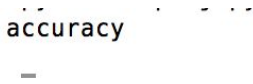
I differentiated the error function using the matrices  $W_{kj}$  and  $W_{ji}$  respectively for calculating differential functions using a set of matrices.

The differentiation of error function with respect to  $W_{kj}$  was to multiply the transpose of  $Z_j$  matrix and the difference of  $Y_k$  and  $T_k$

The differentiation of error function with respect to  $W_{ji}$  first involved determination of a scalar quantity  $H'(j)$  which could be computed by multiplying  $Z_j$  and transpose of  $(1 - Z_j)$

Later, multiplication of the scalar quantity with  $W_{kj}$ ,  $\delta_{ak}$  and  $X_i$  results in obtaining the result of differentiation of error function with  $W_{ji}$ .

Neural networks training accuracy



| Epoch | Accuracy |
|-------|----------|
| 0     | 0.55     |
| 10    | 0.65     |
| 20    | 0.75     |
| 30    | 0.80     |
| 40    | 0.85     |
| 50    | 0.88     |
| 60    | 0.90     |
| 70    | 0.90     |
| 80    | 0.90     |
| 90    | 0.90     |

### Convolutional Neural Network:

The task of this part of the project was to implementing a convolutional neural network for data classification in tensorflow. It involved setting up tensorflow environment in the local system. The stats observed after executing tensor flow are as shown in the figure.

```

[Longclaw:anaconda Manisha$ cd ..
[Longclaw:desktop Manisha$ export PATH=~/.anaconda/bin:$PATH
[Longclaw:desktop Manisha$ source activate tensorflow
[(tensorflow) Longclaw:desktop Manisha$ python tensor.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
step 0, training accuracy 0.12
step 100, training accuracy 0.78
step 200, training accuracy 0.94
step 300, training accuracy 0.86
step 400, training accuracy 0.94
step 500, training accuracy 0.94
step 600, training accuracy 1
step 700, training accuracy 0.98
step 800, training accuracy 0.94
step 900, training accuracy 1
step 1000, training accuracy 0.98
step 1100, training accuracy 1
step 1200, training accuracy 0.98
step 1300, training accuracy 1
step 1400, training accuracy 0.96
step 1500, training accuracy 1
step 1600, training accuracy 0.98
step 1700, training accuracy 1
step 1800, training accuracy 1
step 1900, training accuracy 0.96
test accuracy 0.9727

```

#### Dataset #2: USPS Dataset:

For reading USPS data, we imported Image from PIL and also used glob to read images and the corresponding RGB values were to be noted, hence the values are calculated over a sum of 255.

It was resized to 28\*28 and the results were used as input data.

The MNIST trained data has been compared with USPS data and my findings do support the Free lunch theorem