**REPORT**

**PYTHON VERSION-** Python 3.6 - 64 bit

**Training on 80% data and Testing on 20% data**

**SQUARED\_ERROR USED**

**No\_of\_Features = 8424**

**Cost\_function = Squared\_Error**

**Optimization\_Algorithm=Stochastic Gradient Descent**

**PART (A\_1):**

Here we have used sigmoid function as a activation function in every layer

When we run the code –

1. Graph will be plotted as mentioned in the question
2. Data required to plot this graph will be generated will be written in file
3. Final learned parameter will also be saved in file(W1,W2,W3)

From above plotted graph and data we can easily see with more number of iteration our in sample error become low. I have trained the model 3 iterations using SGD on training data. What I have observed –

3.1361205065989505e-06

7.471167274308915e-07

3.029634798172945e-07

After learned the parameter I have tested 3 iteration on testing data and found out of \_sample error is

0.0

0.0

0.0

And it is fixed and not changing with more no of iteration

Optimal value 3.029634798172945e-07 got in 3rd iteration of training data

Optimal value 0.0 got in 1st iteration of testing data

Here we can see our out\_of\_sample error is less than in\_sample\_error

**Accuracy (in whole data set trained on 80% data)**

Correct - 5573

Incorrect - 1

Total\_Message - 5574

Accuracy - 99.98205956225333

**Accuracy(in whole test data set trained on 80% data)**

correct\_1 - 1115

incorrect\_1 - 0

Total\_Message - 1115

Accuracy\_2 - 100.0

**PART (A\_2):**

Here we have used tanh function as a activation function in every layer

Here we have used sigmoid function as a activation function in every layer

When we run the code –

1. Graph will be plotted as mentioned in the question
2. Data required to plot this graph will be generated will be written in file
3. Final learned parameter will also be saved in file(W1,W2,W3)

From above plotted graph and data we can easily see with more number of iteration our in sample error become low. I have trained the model 3 iterations on training data using SGD . What I have observed –

After 3rd iteration our in sample error becomes fixed like this

2.310980347796598e-27

0.0

1.9721522630525295e-31

After learned the parameter I have tested 3 iteration on testing data and out\_ of \_sample \_error look like this

0.0

0.0

0.0

And it is fixed and not changing with more no of iteration

Optimal value 1.9721522630525295e-31 got in 3rd iteration of training data

Optimal value 0.0 got in 1st iteration of testing data

Here we can see our out\_of\_sample error is less than in\_sample\_error

**Accuracy(in whole data set trained on 80% data)**

correct - 5194

incorrect - 380

Total\_Message - 5574

Accuracy - 93.18263365626122

**Accuracy (in test data set trained on 80% data)**

correct\_1 - 1046

incorrect\_1 - 69

Total\_Message - 1115

Accuracy\_2 - 93.81165919282512

Sigmoid version performs better than tanh version In terms Accuracy in my neural network design

PART(A\_1) performs better than PART(A\_2)

**PART (B1\_1):**

Here we have used sigmoid function as a activation function in every layer except the output layer. In output layer we have used softmax function

When we run the code –

1. Graph will be plotted as mentioned in the question
2. Data required to plot this graph will be generated will be written in file
3. Final learned parameter will also be saved in file(W1,W2,W3)

From above plotted graph and data we can easily see with more number of iteration our in sample error become low. I have trained the model 3 iterations on training data using SGD. What I have observed –

2.3319015997479296e-05

3.157569283728392e-06

1.3461271810880535e-06

After learned the parameter I have tested 3 iteration on testing data and out\_ of \_sample \_error look like this

0.0

0.0

0.0

Optimal value 1.3461271810880535e-06 got in 3rd iteration of training data

Optimal 0.0 got in 1st iteration of testing data

Here we can see our out\_of\_sample error is less than in\_sample\_error

**Accuracy(in whole data set trained on 80% data)**

correct - 5574

incorrect - 0

Total\_Message - 5574

Accuracy - 100.0

**Accuracy(in test data set trained on 80% data)**

correct\_2 - 1115

incorrect\_2 - 0

Total\_Message - 1115

Accuracy\_1 - 100.0

**PART (B\_2):**

Here we have used tanh function as a activation function in every layer except the output layer. In output layer we have used softmax function

When we run the code –

1. Graph will be plotted as mentioned in the question
2. Data required to plot this graph will be generated will be written in file
3. Final learned parameter will also be saved in file(W1,W2,W3)

From above plotted graph and data we can easily see with more number of iteration our in sample error become low. I have trained the model 5 iterations on training data using SGD. What I have observed –

1.7101258791799772e-17

5.9202444746778735e-16

7.210839839909575e-16

1.240123858760157e-15

1.240127387019525e-15

2.4792247640422124e-15

In\_sample\_error is Increasing with Iteration but still it is very less

After learned the parameter I have tested 3 iteration on testing data and out\_ of \_sample \_error look like this

0.0

0.0

0.0

Here we can see our out\_of\_sample error is always less than in\_sample\_error

**Accuracy(in whole data set trained on 80% data)**

Correct - 4967

incorrect - 607

Total\_Message - 5574

Accuracy - 89.11015428776462

**Accuracy(in test data set trained on 80% data)**

correct\_2 - 1001

incorrect\_2 - 114

Total\_Message - 1115

Accuracy\_1- 89.77578475336323

**Final conclusion:** sigmoid activation function(in hidden layer) with softmax function(output layer) performs better than tanh activation function(in hidden layer) with softmax function(output layer) in terms of Accuracy

**PART(B\_1) better than PART(B\_2)**

**AMONG ALL NEURAL NETWORK DESIGN PART(B\_1) PERFORMS BEST BECAUSE IT GIVES 100% ACCURACY IN WHOLE DATA SET AND ALSO TEST DATA SET**

**SECOND BEST NEURAL NETWORK DESIGN IS PART(A\_1)**

**THIRD BEST NEURAL NETWORK DESIGN IS IS PART(A\_2)**

**FOURTH BEST NEURAL NETWORK DESIGN IS IS PART(B\_2)**