6.036 Project 2 WriteUp

Suman Nepal

nsuman@mit.edu

1. Multinomial /Softmax Regression and Gradient Descent

4. The error in the final test is 0.1005

Temperature Parameter

- Having large theta puts more emphasis on the greatest value of the output. So, using lower temp parameter will increase the concentration of probabilities slightly more than with standard but for lower values of thetas, decrease in temperature parameter significantly pronounces the probability for sample with highest output from the regression.
- Error Rate for temp = 0.5 was 0.84
- Error rate for temp = 2 was 0.1261
  Using lower temp parameter concentrated the probabilities to higher values while using higher temp parameter spreads the probabilities resulting in higher error rates. This means that error rate will increase as the temperature parameter increase.

Changing Parameter

0.7001 for mod3 without mod3 training labels

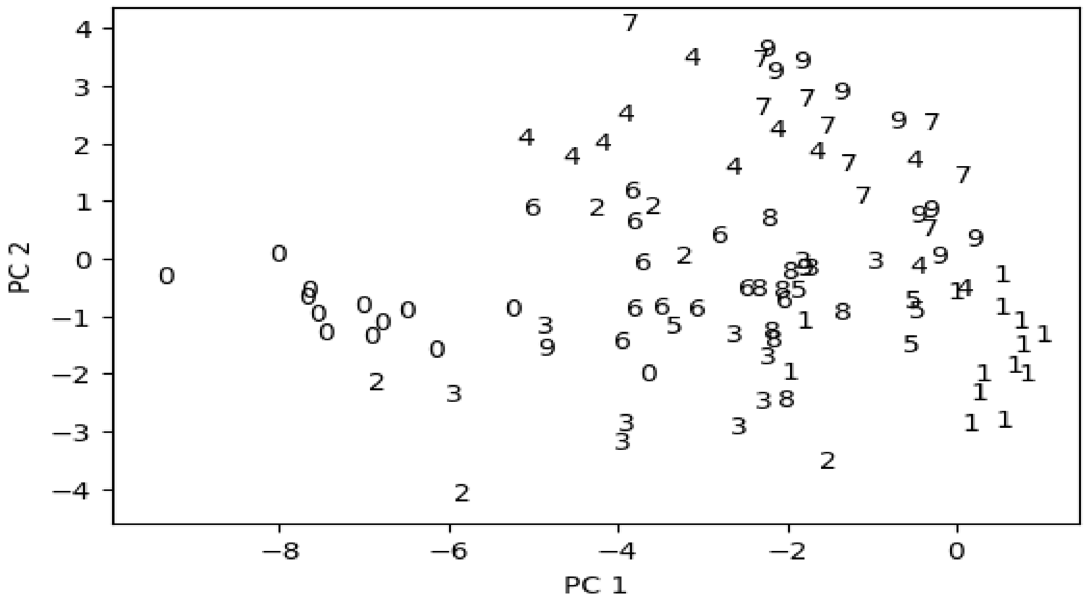
0.1873 for mod3 training labels

Since the neural net trains on the mod value for the second time, the weights of the regression model is trained for giving the mod outputs. This lowers the error rate for the second time. In the first training, without the mod value, we only took the mod value on the output of the model and the model was not optimized for the giving output on the 0-2 scale. So we get the high error rate.

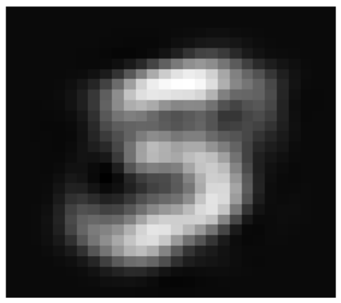
Classification using manually-crafted features

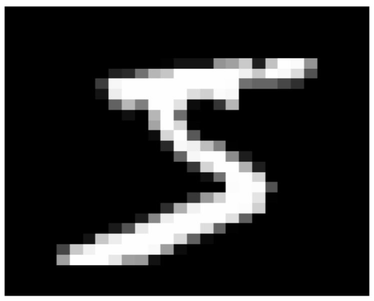
3. Error reported for softmax error for PCA represented data : 0.1439

4.

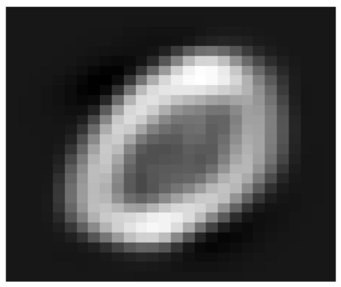


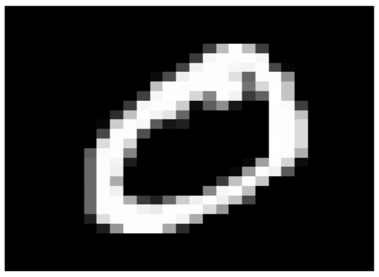
5.





First Original First Principal





Second Original Second Principal

Cubic Features

Phi(x) = [x1^3, x2^3, sqrt(3)*x1^2*x2,sqrt(3)*x1*x2^2, sqrt(3)*x1^2, sqrt(6)*x1*x2, sqrt(3)*x2^2, sqrt(3)*x1, sqrt(3)*x2,1]

7. Error = 0.0865

Neural Networks

learning rate

We may want to decrease the neural network after each epoch. This allows aggressive update in the beginning but delicate exploration of the error surface later.

The generalization in the test set can be low due to over fitting of the data and also the problem of vanishing gradients during training.

The models will increase the accuracy to a certain extent while increasing the number of epochs but reaches a plateau

To find the better model, we can train the network as long as the error is going down. When the error or loss function is no longer decreasing we can stop the training.

4. Classification of MNIST using deep neural Net

1. Fully-Connected Neural Net

(a). 0.9174

(b) I changed the number of layers to 512 and increased the learning rate to 0.1

Decreasing the learning rate didn't work. Adding an additional layer with 128 layers increased the accuracy but not to desired level.

2. Convolutional Neural Net

5. Overlapping

40000 images on train , 10000 on test

42 * 28

The y_train values are the labels corresponding to two different types of number present in the image.

Model.compile compiles the network with the appropriate optimizer, learning rates, and loss function.

Model.fit trains the network in the training data according the parameters specifies.I.e trains the data train_x and train_y till specific number of epochs and using batch_size training sets at a time to update the weights using SGD at a time.

Here the network is trained by Stochastic Gradient descent and the loss function used is categorical crossentropy. The measure used is the accuracy of the classification.

MLP

Mlp.py :

Dense_layer_activation: relu

Output_layer_activation : softmax

Loss: categorical_crossentropy

Optimizer: sgd

Time : 80secs/30 epochs

Accuracy : 0.90

Conv.py

Time: 50secs/epoch for 3 epochs

Accuracy on test set : 97%

Example_model_1:

Dense_layer_activation: relu

Output_layer_activation : softmax

Loss: kullback_leibler_divergence

Optimizer: adam

Time : 180 secs/3 epochs

Accuracy : 0.97

Example_model_2 :

Dense_layer_activation: tanh

Output_layer_activation : softmax

Loss: categorical_crossentropy

Optimizer: adam

Time : 180 secs/3 epochs

Accuracy : 0.95

Example_model_3

Dense_layer_activation: sigmoid

Output_layer_activation : softmax

Loss: categorical_crossentropy

Optimizer: adadelta

Time : 180 secs/3 epochs

Accuracy : 0.93%