Aditya Das

axd5763

1001675762

**Assignment 3**

To run the programs:

Language used: python3

Running problem 1: python3 problem1.py

Running problem2: python3 tf.py

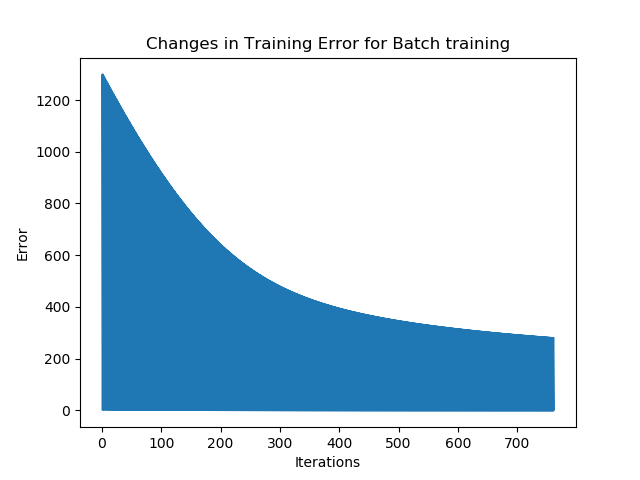
Problem 2 has the hidden node commented by default. To run and change the hidden node, kindly change the snippet at the line.

Problem 1

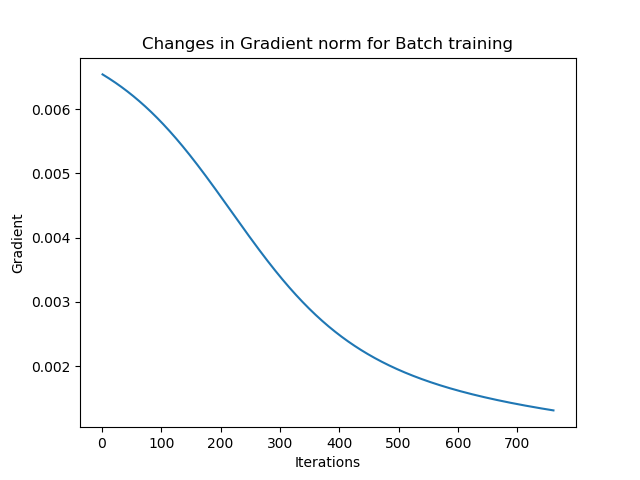
**1.Perform batch training using gradient descent. Divide the derivative with the total number of training dataset as you go through iteration (it is very likely that you will get NaN if you don’t do this.). Change your learning rate as η = {1, 0.1, 0.01, 0.001}. Your report should include: 1) scatter plot of the testing data and the trained decision boundary, 2) figure of changes of training error (cross entropy) w.r.t. iteration, 3) figure of changes of norm of gradient w.r.t. iteration. Also, report the number iterations it took for training and the accuracy that you have.**

**Batch Training:**

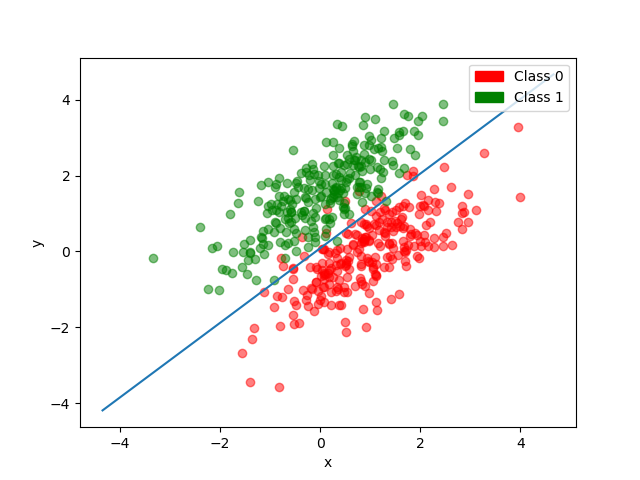
Error plot for learning rate 0.01:



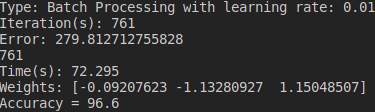
Gradient Plot for learning rate 0.01:



Scatter Plot and Decision boundary for learning rate 0.01:



Batch Training Output:

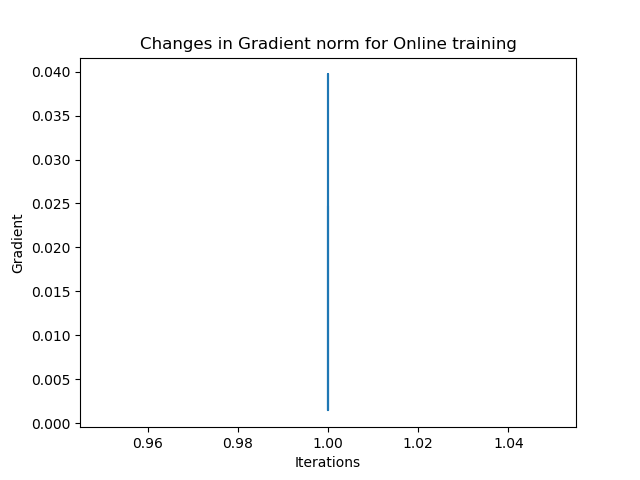


**Online Training:**

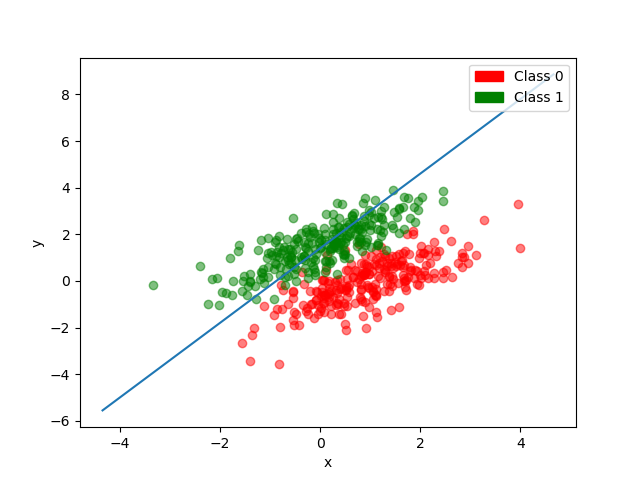
Error plot for learning rate 0.01:



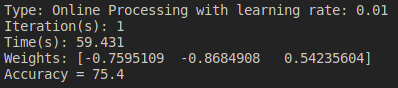
Gradient Plot for learning rate 0.01:



Scatter Plot and Decision boundary for learning rate 0.01:

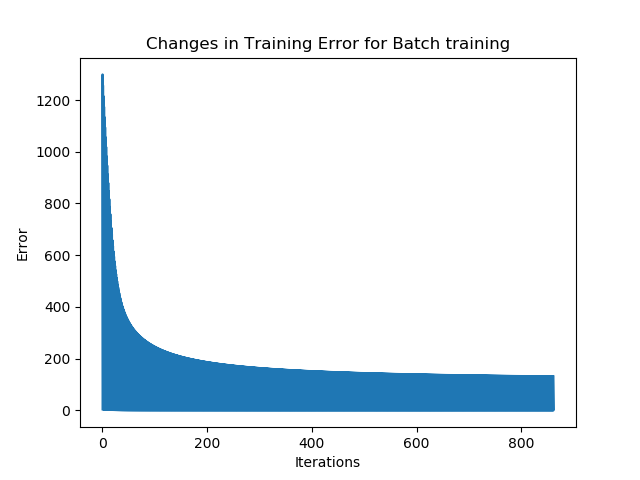


Online Training Output:

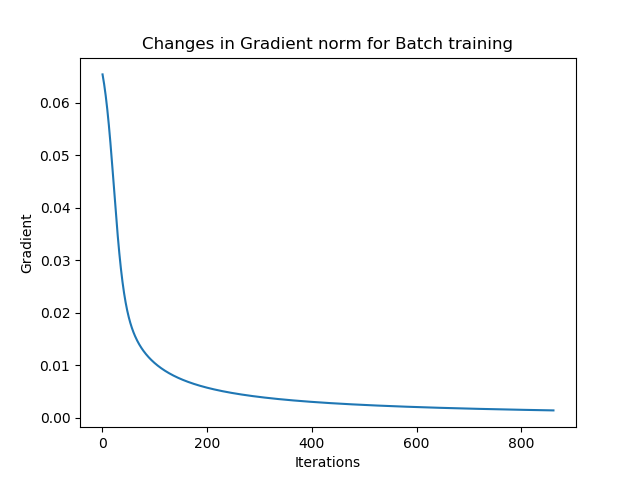


**Batch Training:**

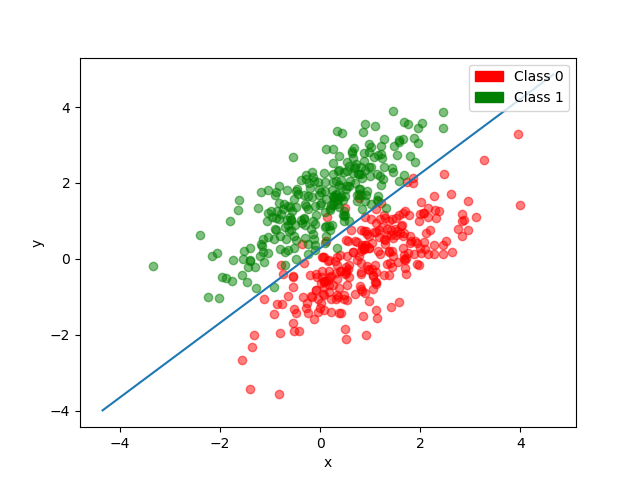
Error plot for learning rate 0.1:



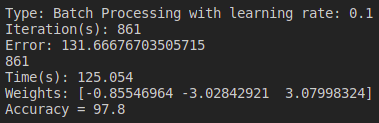
Gradient Plot for learning rate 0.1:



Scatter Plot and Decision boundary for learning rate 0.1:



Batch Training Output:

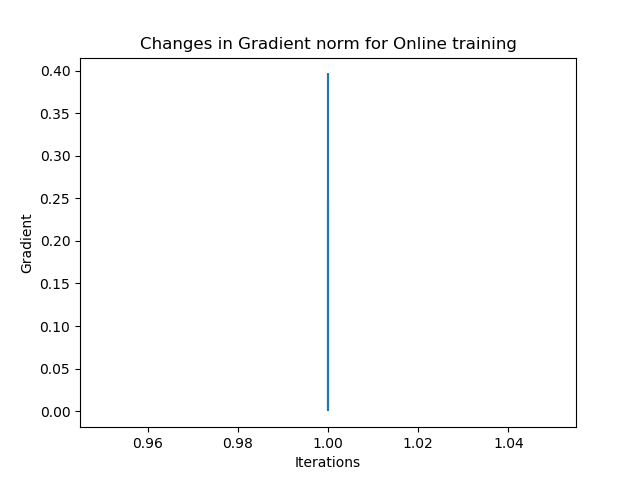


**Online Training:**

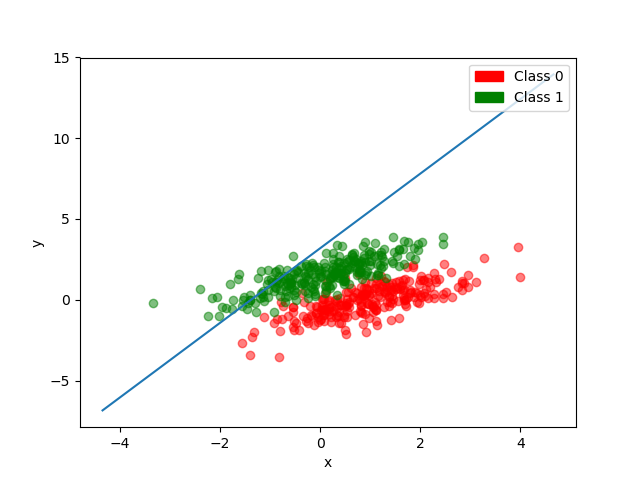
Error plot for learning rate 0.1:



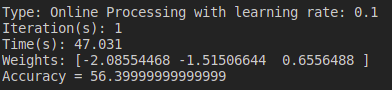
Gradient Plot for learning rate 0.1:



Scatter Plot and Decision boundary for learning rate 0.1:

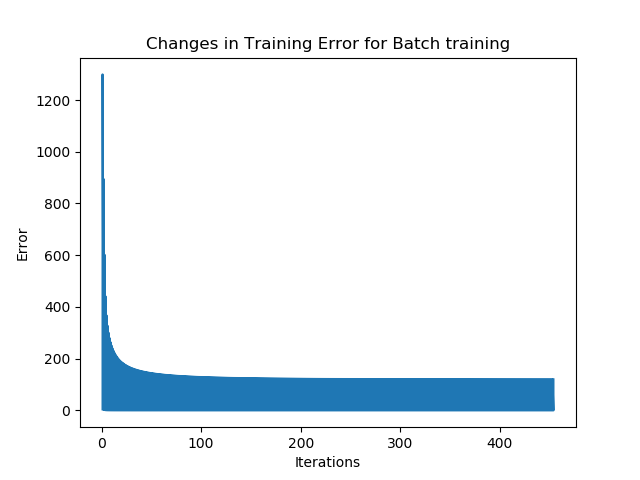


Online Training Output:

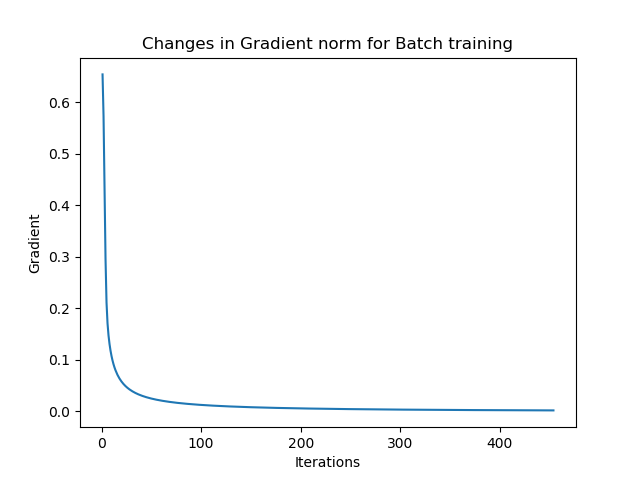


**Batch Training:**

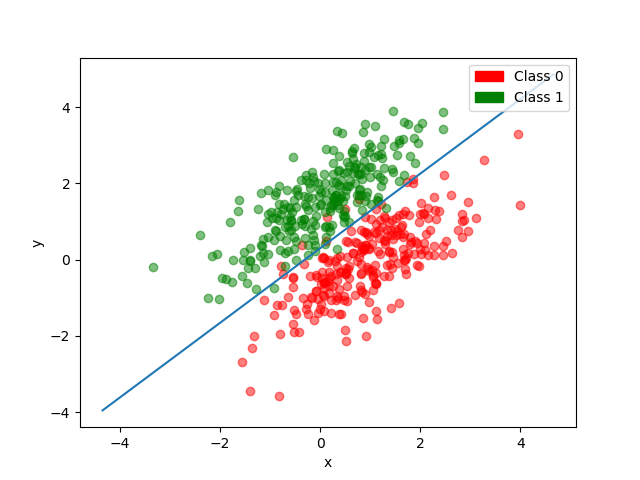
Error plot for learning rate 1:



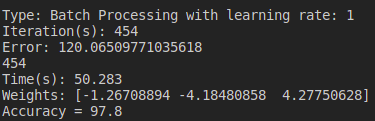
Gradient Plot for learning rate 1:



Scatter Plot and Decision boundary for learning rate 1:

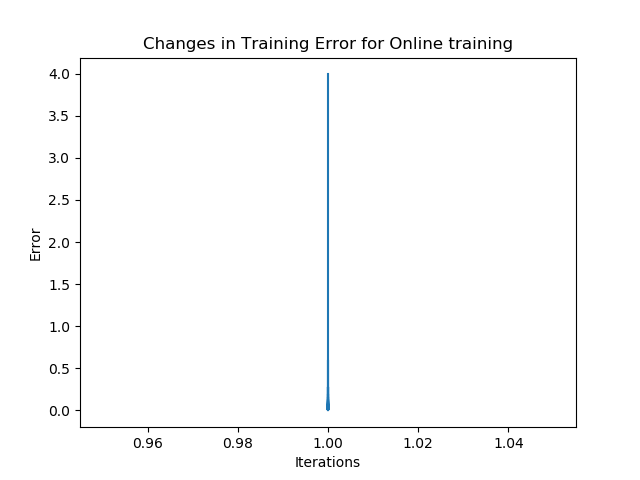


Batch Training Output:

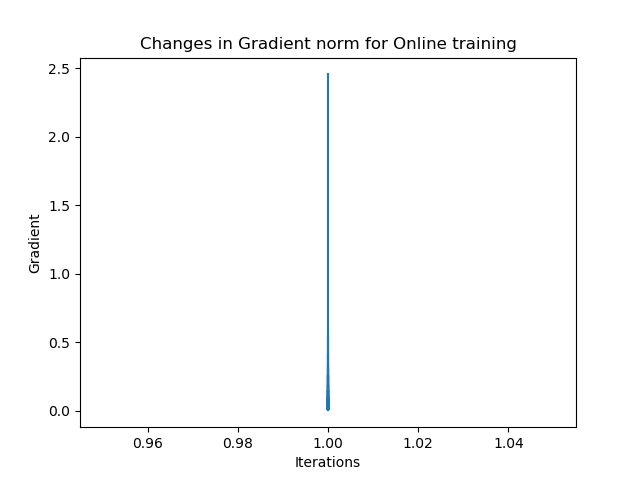


**Online Training:**

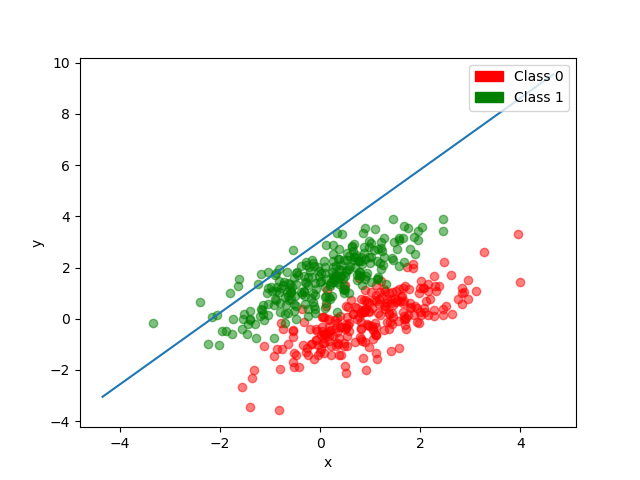
Error plot for learning rate 1:



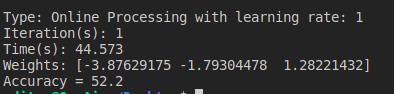
Gradient Plot for learning rate 1:



Scatter Plot and Decision boundary for learning rate 1:



Online Training Output:



Problem 2

1. import tensorflow as tf

2. mnist = tf.keras.datasets.mnist

3.

4. (x\_train, y\_train),(x\_test, y\_test) = mnist.load\_data()

5. x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

6.

7. model = tf.keras.models.Sequential([

8. tf.keras.layers.Flatten(),

9.tf.keras.layers.Dense(512, activation=tf.nn.relu),

10. tf.keras.layers.Dropout(0.2),

11. tf.keras.layers.Dense(10, activation=tf.nn.softmax)

12. ])

13.

14. model.compile(

15. optimizer='adam',

16. loss='sparse\_categorical\_crossentropy',

17. metrics=['accuracy'] )

18.

19. model.fit(x\_train, y\_train, epochs=5)

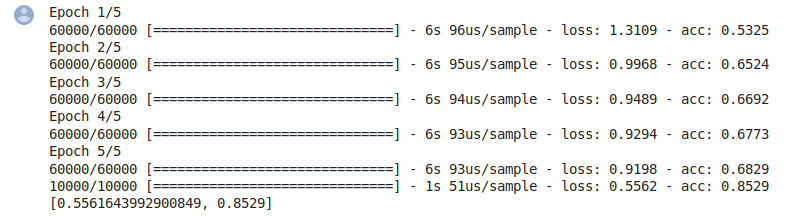
20. model.evaluate(x\_test, y\_test)

**1.** **In the report, write comments for each line of code given above and explain what this framework is doing.**

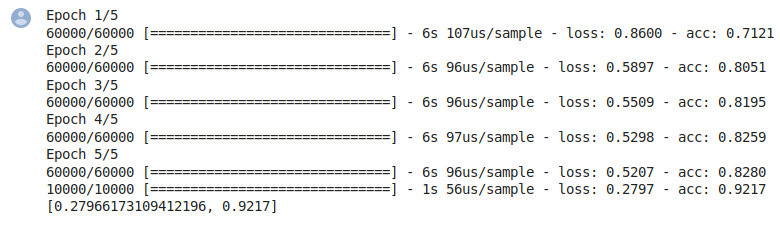
|  |  |
| --- | --- |
| **Line no.** | **Comments** |
| 1 | Importing the **tensorflow** library under the alias **tf** to use in the program |
| 2 | importing the **mnist** dataset which consists of image of **handwritten digits** of size 28\*28 pixels. |
| 4 | Loading the **mnist** training data and testing data of size **6000** and **1000** respectively.  x\_train and x\_test represents the matrix of 28\*28, while y\_train and y\_test is the output. |
| 5 | **Normalizing** the dataset with values between 0 to 1 |
| 7 | **tf.keras.models.Sequential** creates a deep learning model with linear stack of layers. |
| 8 | tf.keras.layers.Flatten() Flattens the input. |
| 9 | **tf.keras.layers.Dense(512, activation = tf.nn.relu)** This creates a layer in the network which has 512 nodes and all with the activation function of ‘Rectified Linear Unit’. |
| 10 | **tf.keras.layers.Dropout(0.2)** This helps to evaluate the network better by droping out 20% of  nodes for each iterations. This helps to deviate the model from over fitting. |
| 11 | **tf.keras.layers.Dense(10, activation = tf.nn.softmax)** This creates a layer in the network which has 10 nodes and all with the activation function of ‘Softmax’. Since this is the last layer, we can say the same as output layer. |
| 14 | **model.compile()** helps us to add configurations to the model. They include learning rates, error  rate methods, different optimizer types and many more. |
| 15 | Model to use Adam Optimizer Algorithm |
| 16 | Model to use Sparse Categorial Cross Entropy to calculate the loss |
| 17 | Metrics to be evaluated by the model during training and testing. Here, accuracy |
| 19 | **model.fit(x\_train, y\_train, epochs=5)** helps to train the model with input, required output.  Here epochs is number of batch iterations. |
| 20 | **Model.evaluate(x\_test, y\_test)** helps to test the model with input and required output. |

**2. Change the number of hidden nodes to 5, 10, 128 and 512. Report how the testing accuracy changes for the testing data. Report the result and your observation in the report.**

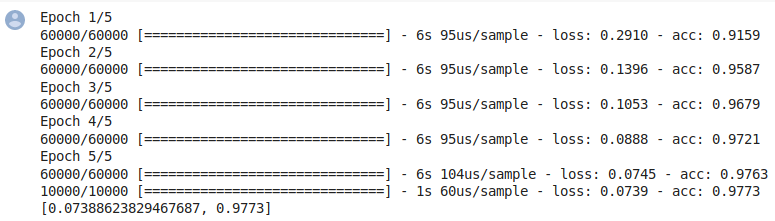
Hidden nodes 5



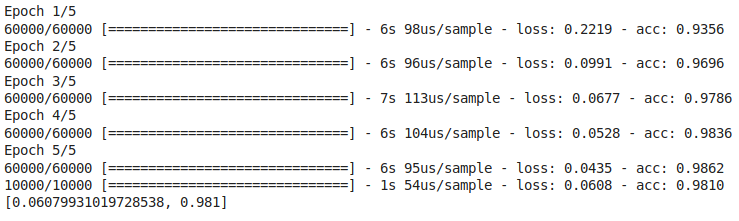
Hidden notes 10



Hidden notes 128



Hidden notes 512



**3.** **Now, remove the hidden layer in the code and train the model. The trained model contains the weights that it has learned from training. Plot the “new representation” that it has learned for each number from training and include them in the report. That is, reshape the learned weights (i.e.,vector) to the image dimension (in 2D, i.e., 28x28) and show them. You will see some number-like features.**

The program was changed, and the following code was added.

a = model.get\_weights()[0]

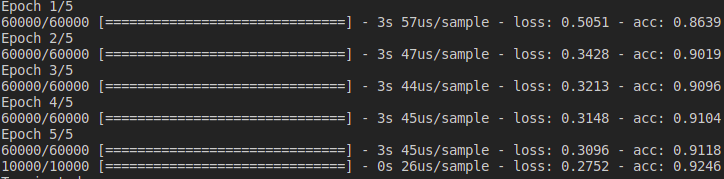
for i in range(0,10):

b = tf.reshape(a[:,i], [28,28]) # Choosing column i to procude 28x28 color matrix

plt.title("Image " + str(i) + ":")

plt.imshow(tf.Session().run(b), cmap='gray')

plt.show()



The images are attached in a separate folder under the subfolder problem