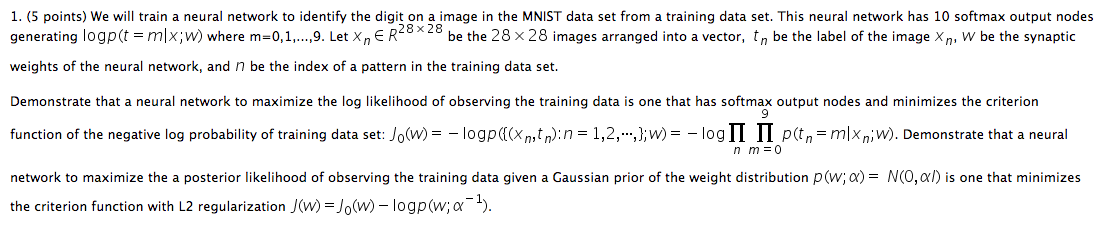
CSE 455-555

Assignment 4-5 Report

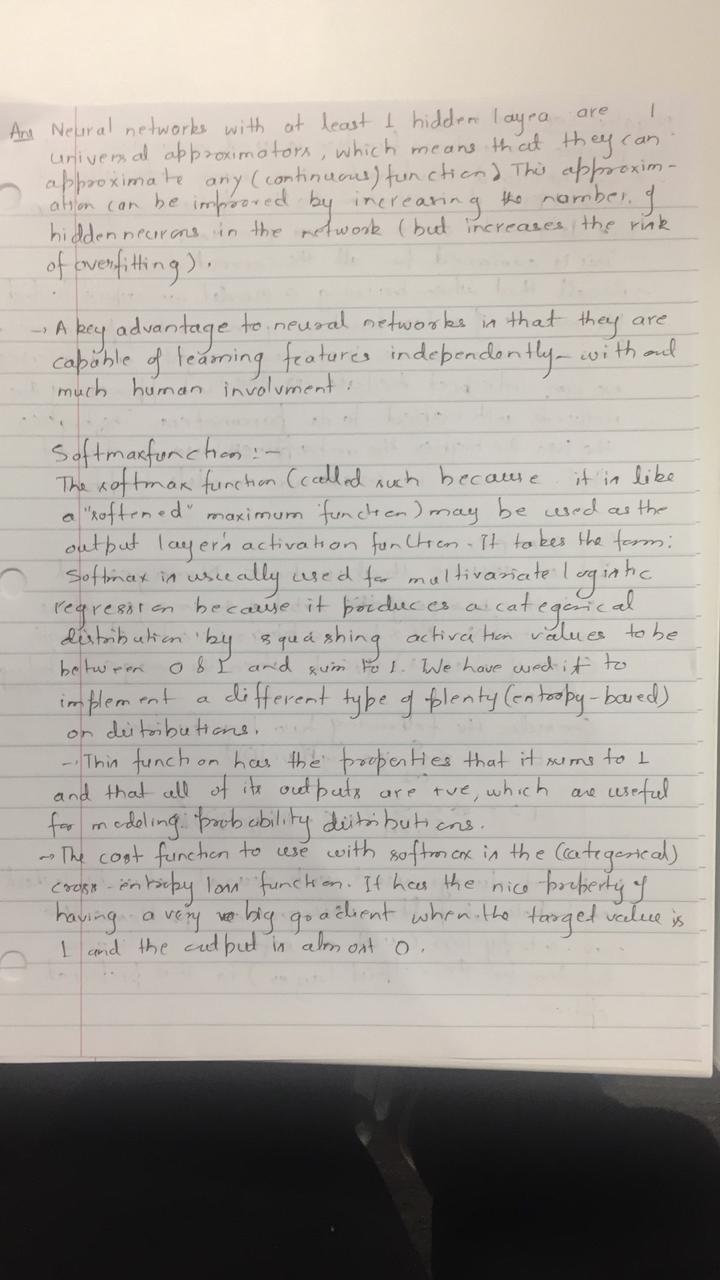
Name: Rajiv Ranjan

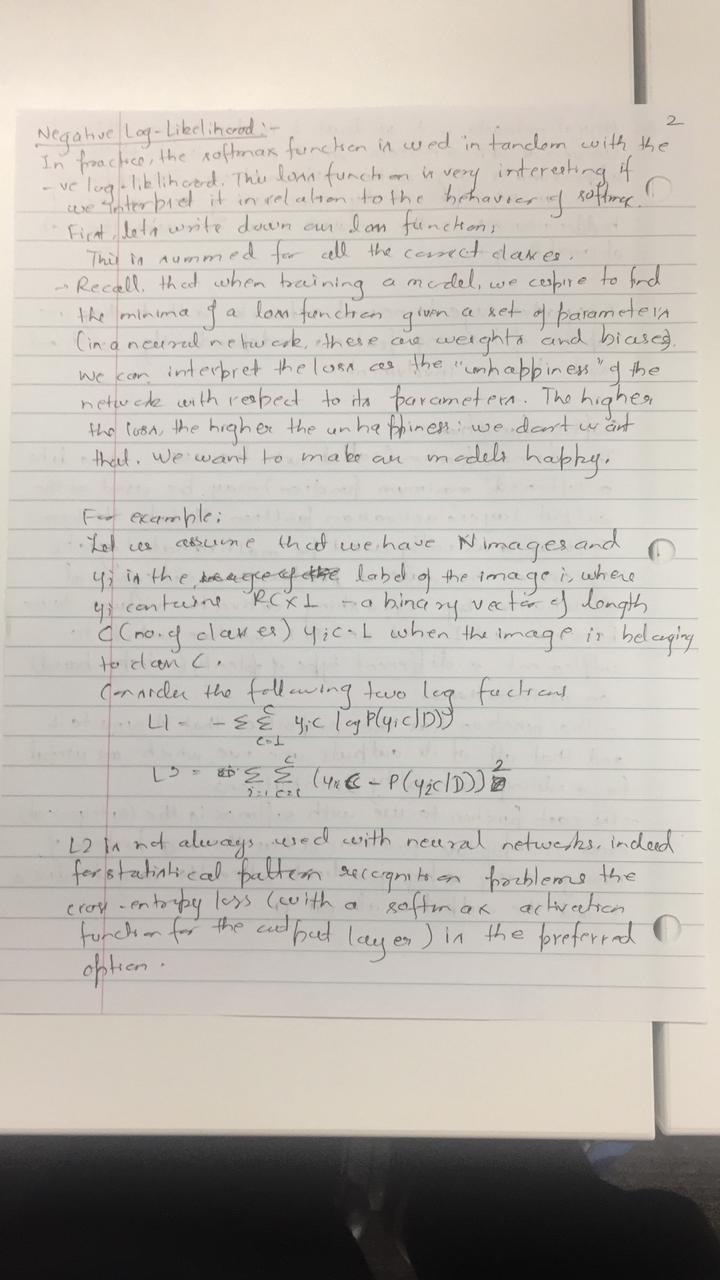
Ub id: 50249099

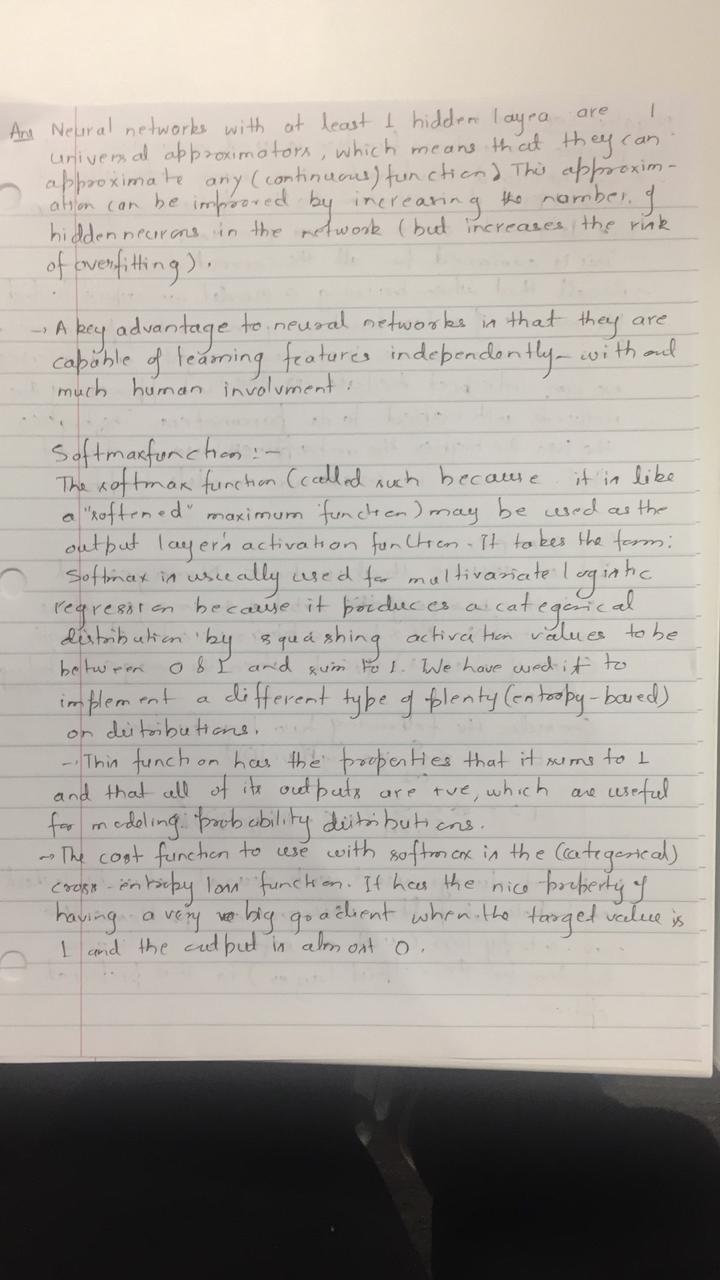
Ub name: rajivran



Ans:







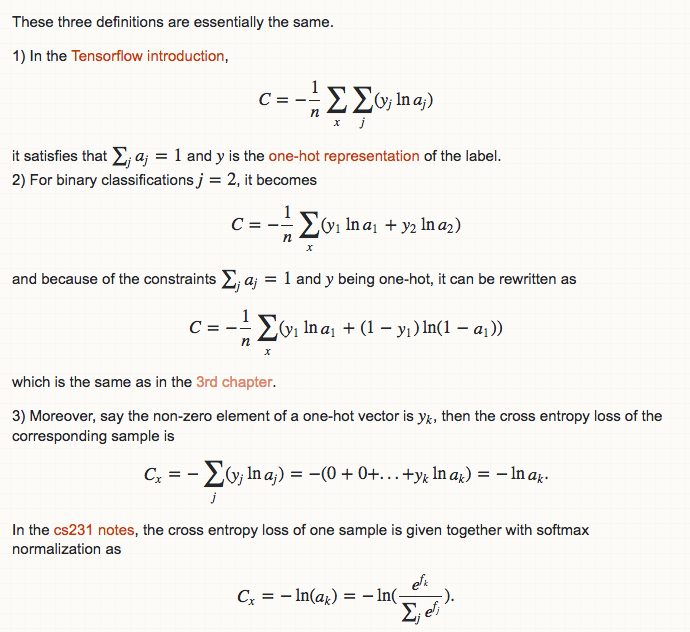
Other than the provided answer the below claims also support the argument:

The negative log likelihood (eq.80) is also known as the multiclass cross-entropy (ref: Pattern Recognition and Machine Learning Section 4.3.4), as they are in fact two different interpretations of the same formula.

eq.57 is the negative log likelihood of the Bernoulli distribution, whereas eq.80 is the negative log likelihood of the multinomial distribution with one observation (a multiclass version of Bernoulli).

For binary classification problems, the softmax function outputs two values (between 0 and 1 and sum to 1) to give the prediction of each class. While the sigmoid function outputs one value (between 0 and 1) to give the prediction of one class (so the other class is 1-p).

So eq.80 can't be directly applied to the sigmoid output, though it is essentially the same loss as eq.57.



P.s: the following equation has been taken from a blog on neural networks, found online.

2 (a). (5 points) Build a neural network with 1 hidden layer of 30 sigmoid nodes, and an output layer10 softmax nodes from 1000 training images (100 images per digit). Train the network for 30 complete epochs, using mini-batches of 10 training examples at a time, a learning rate η=0.1. Plot the training error, testing error, criterion function on training data set, criterion function on testing data set of a separate 1000 testing images (100 images per digit), and the learning speed of the hidden layer (the average absolute changes of weights divided by the values of the weights).

Soln:

The code for this is the following file:single\_hidden\_layer

Here are the specifications of this execution:

The following libraries have been used: tensorflow along with keras.

Also the following requirements have been met in the code:

1. The code was run only on 1000 training and 1000 test images.
2. It had only one hidden layer.
3. The hidden layer had sigmoid as it’s activation function.
4. There were 30 sigmoid nodes.
5. An output layer was there having 10 softmax nodes.
6. The network was trained for 30 complete epochs.
7. Also a mini batch of 10 training examples was used at a time
8. Learning rate was in initially kept as 0.1
9. In keras the learning rate speed can be changed using the decay parameter of the activation function

Below is the output:

Layer (type) Output Shape Param #

=================================================================

dense\_3 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 10) 310

=================================================================

Total params: 23,860

Trainable params: 23,860

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

/Users/rajivranjan/anaconda3/lib/python3.6/site-packages/keras/models.py:942: UserWarning: The `nb\_epoch` argument in `fit` has been renamed `epochs`.

warnings.warn('The `nb\_epoch` argument in `fit` '

- 0s - loss: 2.2700 - acc: 0.1830 - val\_loss: 2.1821 - val\_acc: 0.4040

- LR: 0.090909

Epoch 2/30

- 0s - loss: 2.0282 - acc: 0.4160 - val\_loss: 1.8824 - val\_acc: 0.5900

- LR: 0.083333

Epoch 3/30

- 0s - loss: 1.6315 - acc: 0.6300 - val\_loss: 1.5300 - val\_acc: 0.5910

- LR: 0.076923

Epoch 4/30

- 0s - loss: 1.2944 - acc: 0.7160 - val\_loss: 1.2702 - val\_acc: 0.6580

- LR: 0.071429

Epoch 5/30

- 0s - loss: 1.0616 - acc: 0.7660 - val\_loss: 1.1003 - val\_acc: 0.7110

- LR: 0.066667

Epoch 6/30

- 0s - loss: 0.8979 - acc: 0.8050 - val\_loss: 0.9914 - val\_acc: 0.7340

- LR: 0.062500

Epoch 7/30

- 0s - loss: 0.7893 - acc: 0.8270 - val\_loss: 0.8957 - val\_acc: 0.7540

- LR: 0.058824

Epoch 8/30

- 0s - loss: 0.7050 - acc: 0.8470 - val\_loss: 0.8329 - val\_acc: 0.7620

- LR: 0.055556

Epoch 9/30

- 0s - loss: 0.6441 - acc: 0.8520 - val\_loss: 0.7924 - val\_acc: 0.7790

- LR: 0.052632

Epoch 10/30

- 0s - loss: 0.5927 - acc: 0.8660 - val\_loss: 0.7525 - val\_acc: 0.7810

- LR: 0.050000

Epoch 11/30

- 0s - loss: 0.5545 - acc: 0.8730 - val\_loss: 0.7189 - val\_acc: 0.7940

- LR: 0.047619

Epoch 12/30

- 0s - loss: 0.5210 - acc: 0.8760 - val\_loss: 0.6961 - val\_acc: 0.8000

- LR: 0.045455

Epoch 13/30

- 0s - loss: 0.4930 - acc: 0.8840 - val\_loss: 0.6772 - val\_acc: 0.8000

- LR: 0.043478

Epoch 14/30

- 0s - loss: 0.4687 - acc: 0.8900 - val\_loss: 0.6615 - val\_acc: 0.8010

- LR: 0.041667

Epoch 15/30

- 0s - loss: 0.4477 - acc: 0.8970 - val\_loss: 0.6465 - val\_acc: 0.8100

- LR: 0.040000

Epoch 16/30

- 0s - loss: 0.4291 - acc: 0.8950 - val\_loss: 0.6342 - val\_acc: 0.8070

- LR: 0.038462

Epoch 17/30

- 0s - loss: 0.4138 - acc: 0.9060 - val\_loss: 0.6201 - val\_acc: 0.8110

- LR: 0.037037

Epoch 18/30

- 0s - loss: 0.3986 - acc: 0.9070 - val\_loss: 0.6111 - val\_acc: 0.8130

- LR: 0.035714

Epoch 19/30

- 0s - loss: 0.3859 - acc: 0.9140 - val\_loss: 0.6063 - val\_acc: 0.8110

- LR: 0.034483

Epoch 20/30

- 0s - loss: 0.3740 - acc: 0.9140 - val\_loss: 0.5941 - val\_acc: 0.8130

- LR: 0.033333

Epoch 21/30

- 0s - loss: 0.3638 - acc: 0.9170 - val\_loss: 0.5847 - val\_acc: 0.8210

- LR: 0.032258

Epoch 22/30

- 0s - loss: 0.3536 - acc: 0.9220 - val\_loss: 0.5796 - val\_acc: 0.8200

- LR: 0.031250

Epoch 23/30

- 0s - loss: 0.3440 - acc: 0.9270 - val\_loss: 0.5730 - val\_acc: 0.8280

- LR: 0.030303

Epoch 24/30

- 0s - loss: 0.3361 - acc: 0.9250 - val\_loss: 0.5663 - val\_acc: 0.8250

- LR: 0.029412

Epoch 25/30

- 0s - loss: 0.3282 - acc: 0.9280 - val\_loss: 0.5633 - val\_acc: 0.8280

- LR: 0.028571

Epoch 26/30

- 0s - loss: 0.3213 - acc: 0.9280 - val\_loss: 0.5594 - val\_acc: 0.8270

- LR: 0.027778

Epoch 27/30

- 0s - loss: 0.3145 - acc: 0.9290 - val\_loss: 0.5534 - val\_acc: 0.8280

- LR: 0.027027

Epoch 28/30

- 0s - loss: 0.3077 - acc: 0.9340 - val\_loss: 0.5493 - val\_acc: 0.8310

- LR: 0.026316

Epoch 29/30

- 0s - loss: 0.3022 - acc: 0.9350 - val\_loss: 0.5458 - val\_acc: 0.8280

- LR: 0.025641

Epoch 30/30

- 0s - loss: 0.2966 - acc: 0.9350 - val\_loss: 0.5451 - val\_acc: 0.8260

- LR: 0.025000

Baseline Error: 17.40%

dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])

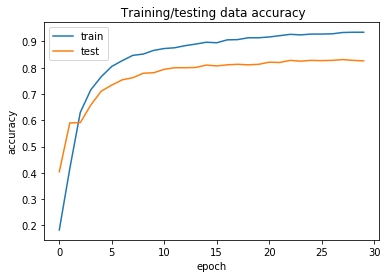
As we can see the accuracy is around 83 percent on test data, 94 on training data and error is around 17 percent.

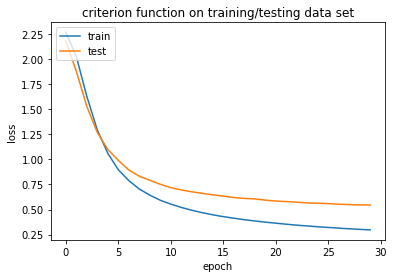
These are good metrics considering we used only 1000 images for our implementation, and it is well known that for neural networks at least we need millions of images to get a good model.

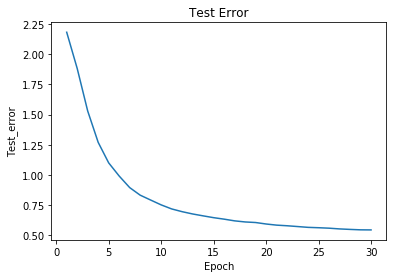
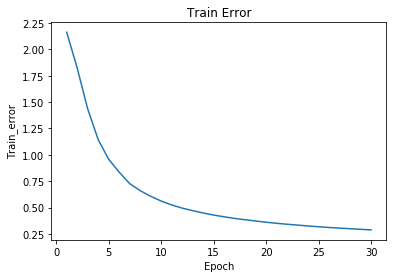
2(a): part 2)

Plot the training error, testing error, criterion function on training data set, criterion function on testing data set of a separate 1000 testing images (100 images per digit), and the learning speed of the hidden layer (the average absolute changes of weights divided by the values of the weights).

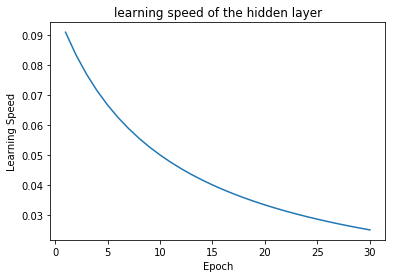
Soln:

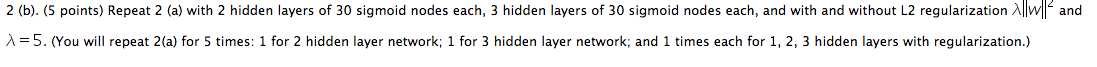






For calculating the learning speed the weights were calculated at each step. They are not shown here because they would take a lot of space.





Soln:

Below are the five files for this question:

two\_hidden\_layer.py--- two\_hidden\_layer without the L2 regularization

three\_hidden\_layer.py---three hidden layer without the L2 regularization

single\_layer\_l2.py --- single layer with L2 regularization, lambda=5

two\_hidden\_l2.py----two hidden layer with L2 regularization, lambda=5

three\_hidden\_l2.py ---- three hidden layer with L2 regularization, lambda=5

output of two\_hidden\_layer.py--- two\_hidden\_layer without the L2 regularization :

Layer (type) Output Shape Param #

=================================================================

dense\_5 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 10) 310

=================================================================

Total params: 24,790

Trainable params: 24,790

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/Users/rajivranjan/anaconda3/lib/python3.6/site-packages/keras/models.py:942: UserWarning: The `nb\_epoch` argument in `fit` has been renamed `epochs`.

warnings.warn('The `nb\_epoch` argument in `fit` '

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

- 0s - loss: 2.3276 - acc: 0.0860 - val\_loss: 2.3050 - val\_acc: 0.1000

- LR: 0.099900

Epoch 2/30

- 0s - loss: 2.3274 - acc: 0.0860 - val\_loss: 2.3023 - val\_acc: 0.1000

- LR: 0.099800

Epoch 3/30

- 0s - loss: 2.3234 - acc: 0.0760 - val\_loss: 2.3074 - val\_acc: 0.1000

- LR: 0.099701

Epoch 4/30

- 0s - loss: 2.3163 - acc: 0.1050 - val\_loss: 2.3121 - val\_acc: 0.1000

- LR: 0.099602

Epoch 5/30

- 0s - loss: 2.3197 - acc: 0.0740 - val\_loss: 2.3150 - val\_acc: 0.1000

- LR: 0.099502

Epoch 6/30

- 0s - loss: 2.3120 - acc: 0.1020 - val\_loss: 2.3157 - val\_acc: 0.1000

- LR: 0.099404

Epoch 7/30

- 0s - loss: 2.3113 - acc: 0.1010 - val\_loss: 2.2981 - val\_acc: 0.2000

- LR: 0.099305

Epoch 8/30

- 0s - loss: 2.2979 - acc: 0.1200 - val\_loss: 2.3027 - val\_acc: 0.1000

- LR: 0.099206

Epoch 9/30

- 0s - loss: 2.2922 - acc: 0.1230 - val\_loss: 2.2810 - val\_acc: 0.1160

- LR: 0.099108

Epoch 10/30

- 0s - loss: 2.2695 - acc: 0.1590 - val\_loss: 2.2510 - val\_acc: 0.3130

- LR: 0.099010

Epoch 11/30

- 0s - loss: 2.2224 - acc: 0.2380 - val\_loss: 2.1829 - val\_acc: 0.2680

- LR: 0.098912

Epoch 12/30

- 0s - loss: 2.1192 - acc: 0.2810 - val\_loss: 2.0821 - val\_acc: 0.2720

- LR: 0.098814

Epoch 13/30

- 0s - loss: 1.9850 - acc: 0.3000 - val\_loss: 1.9313 - val\_acc: 0.3510

- LR: 0.098717

Epoch 14/30

- 0s - loss: 1.8203 - acc: 0.3570 - val\_loss: 1.7833 - val\_acc: 0.4370

- LR: 0.098619

Epoch 15/30

- 0s - loss: 1.6430 - acc: 0.4190 - val\_loss: 1.6186 - val\_acc: 0.4290

- LR: 0.098522

Epoch 16/30

- 0s - loss: 1.4836 - acc: 0.4680 - val\_loss: 1.4931 - val\_acc: 0.4740

- LR: 0.098425

Epoch 17/30

- 0s - loss: 1.3652 - acc: 0.5280 - val\_loss: 1.4035 - val\_acc: 0.5040

- LR: 0.098328

Epoch 18/30

- 0s - loss: 1.2758 - acc: 0.5620 - val\_loss: 1.3373 - val\_acc: 0.5580

- LR: 0.098232

Epoch 19/30

- 0s - loss: 1.1999 - acc: 0.6110 - val\_loss: 1.2843 - val\_acc: 0.5510

- LR: 0.098135

Epoch 20/30

- 0s - loss: 1.1305 - acc: 0.6290 - val\_loss: 1.2458 - val\_acc: 0.5630

- LR: 0.098039

Epoch 21/30

- 0s - loss: 1.0697 - acc: 0.6780 - val\_loss: 1.1888 - val\_acc: 0.6120

- LR: 0.097943

Epoch 22/30

- 0s - loss: 1.0111 - acc: 0.6900 - val\_loss: 1.1489 - val\_acc: 0.6200

- LR: 0.097847

Epoch 23/30

- 0s - loss: 0.9568 - acc: 0.7320 - val\_loss: 1.0907 - val\_acc: 0.6600

- LR: 0.097752

Epoch 24/30

- 0s - loss: 0.8968 - acc: 0.7480 - val\_loss: 1.0723 - val\_acc: 0.6430

- LR: 0.097656

Epoch 25/30

- 0s - loss: 0.8435 - acc: 0.7680 - val\_loss: 1.0196 - val\_acc: 0.6770

- LR: 0.097561

Epoch 26/30

- 0s - loss: 0.7967 - acc: 0.7780 - val\_loss: 0.9871 - val\_acc: 0.6970

- LR: 0.097466

Epoch 27/30

- 0s - loss: 0.7526 - acc: 0.7990 - val\_loss: 0.9554 - val\_acc: 0.7060

- LR: 0.097371

Epoch 28/30

- 0s - loss: 0.7097 - acc: 0.8140 - val\_loss: 0.9253 - val\_acc: 0.7130

- LR: 0.097276

Epoch 29/30

- 0s - loss: 0.6722 - acc: 0.8160 - val\_loss: 0.9065 - val\_acc: 0.7220

- LR: 0.097182

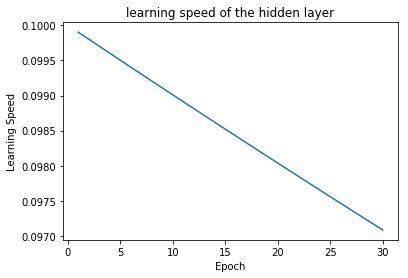
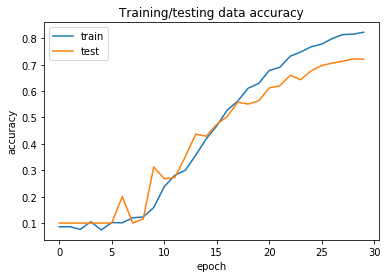
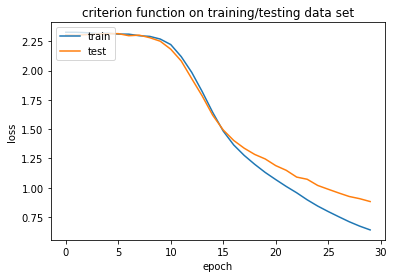
Epoch 30/30

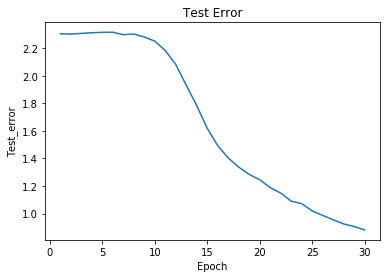
- 0s - loss: 0.6397 - acc: 0.8230 - val\_loss: 0.8820 - val\_acc: 0.7210

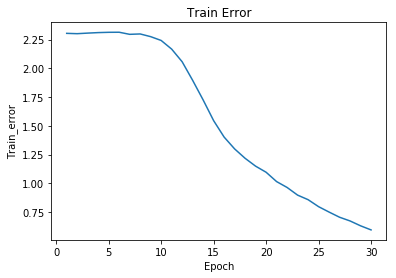
- LR: 0.097087

Baseline Error: 27.90%

dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])







output of three\_hidden\_layer.py--- three\_hidden\_layer without the L2 regularization:

Layer (type) Output Shape Param #

=================================================================

dense\_36 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_37 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_38 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_39 (Dense) (None, 10) 310

=================================================================

Total params: 25,720

Trainable params: 25,720

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/Users/rajivranjan/anaconda3/lib/python3.6/site-packages/keras/models.py:942: UserWarning: The `nb\_epoch` argument in `fit` has been renamed `epochs`.

warnings.warn('The `nb\_epoch` argument in `fit` '

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

- 1s - loss: 2.3134 - acc: 0.0830 - val\_loss: 2.3027 - val\_acc: 0.1000

- LR: 0.009091

Epoch 2/30

- 0s - loss: 2.3053 - acc: 0.0720 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.004762

Epoch 3/30

- 0s - loss: 2.3041 - acc: 0.0760 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.003226

Epoch 4/30

- 0s - loss: 2.3037 - acc: 0.0930 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.002439

Epoch 5/30

- 0s - loss: 2.3035 - acc: 0.0840 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.001961

Epoch 6/30

- 0s - loss: 2.3033 - acc: 0.0860 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.001639

Epoch 7/30

- 0s - loss: 2.3032 - acc: 0.0880 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.001408

Epoch 8/30

- 0s - loss: 2.3031 - acc: 0.0850 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.001235

Epoch 9/30

- 0s - loss: 2.3030 - acc: 0.0790 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.001099

Epoch 10/30

- 0s - loss: 2.3030 - acc: 0.0760 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000990

Epoch 11/30

- 0s - loss: 2.3029 - acc: 0.0830 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000901

Epoch 12/30

- 0s - loss: 2.3029 - acc: 0.0870 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000826

Epoch 13/30

- 0s - loss: 2.3029 - acc: 0.0780 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000763

Epoch 14/30

- 0s - loss: 2.3029 - acc: 0.0830 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000709

Epoch 15/30

- 0s - loss: 2.3028 - acc: 0.0970 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000662

Epoch 16/30

- 0s - loss: 2.3028 - acc: 0.0980 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000621

Epoch 17/30

- 0s - loss: 2.3028 - acc: 0.1000 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000585

Epoch 18/30

- 0s - loss: 2.3028 - acc: 0.0900 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000552

Epoch 19/30

- 0s - loss: 2.3028 - acc: 0.0810 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000524

Epoch 20/30

- 0s - loss: 2.3028 - acc: 0.0840 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000498

Epoch 21/30

- 0s - loss: 2.3028 - acc: 0.0720 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000474

Epoch 22/30

- 0s - loss: 2.3028 - acc: 0.0890 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000452

Epoch 23/30

- 0s - loss: 2.3027 - acc: 0.0940 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000433

Epoch 24/30

- 0s - loss: 2.3027 - acc: 0.0870 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000415

Epoch 25/30

- 0s - loss: 2.3027 - acc: 0.0760 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000398

Epoch 26/30

- 0s - loss: 2.3027 - acc: 0.0920 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000383

Epoch 27/30

- 0s - loss: 2.3027 - acc: 0.0950 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000369

Epoch 28/30

- 0s - loss: 2.3027 - acc: 0.0890 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000356

Epoch 29/30

- 0s - loss: 2.3027 - acc: 0.0980 - val\_loss: 2.3026 - val\_acc: 0.1000

- LR: 0.000344

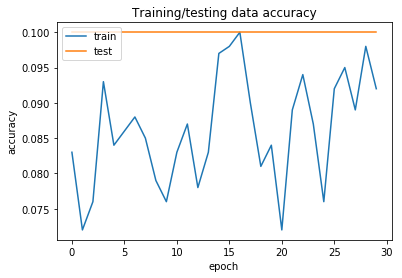
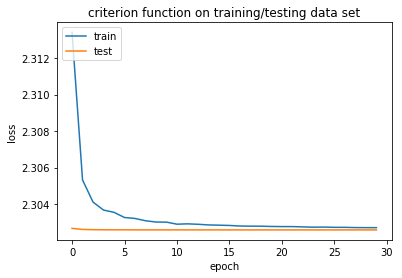
Epoch 30/30

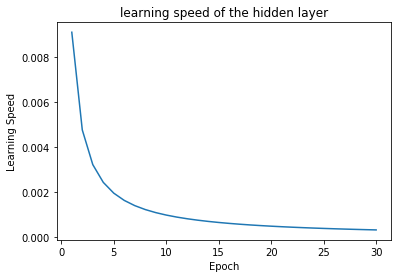
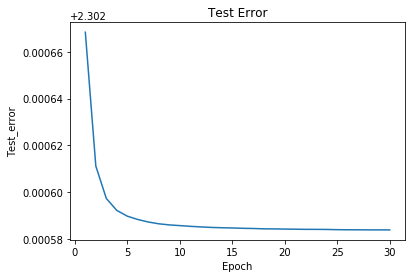
- 0s - loss: 2.3027 - acc: 0.0920 - val\_loss: 2.3026 - val\_acc: 0.1000

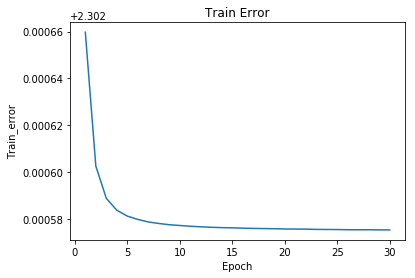
- LR: 0.000332

Baseline Error: 10.00%

dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])







single\_layer\_l2.py --- single layer with L2 regularization, lambda=5

output:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_40 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_41 (Dense) (None, 10) 310

=================================================================

Total params: 23,860

Trainable params: 23,860

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

- 1s - loss: 2.2759 - acc: 0.1830 - val\_loss: 2.1882 - val\_acc: 0.4040

- LR: 0.090909

Epoch 2/30

- 0s - loss: 2.0348 - acc: 0.4160 - val\_loss: 1.8895 - val\_acc: 0.5900

- LR: 0.083333

Epoch 3/30

- 0s - loss: 1.6393 - acc: 0.6300 - val\_loss: 1.5381 - val\_acc: 0.5910

- LR: 0.076923

Epoch 4/30

- 0s - loss: 1.3030 - acc: 0.7150 - val\_loss: 1.2791 - val\_acc: 0.6580

- LR: 0.071429

Epoch 5/30

- 0s - loss: 1.0709 - acc: 0.7660 - val\_loss: 1.1096 - val\_acc: 0.7110

- LR: 0.066667

Epoch 6/30

- 0s - loss: 0.9075 - acc: 0.8050 - val\_loss: 1.0011 - val\_acc: 0.7350

- LR: 0.062500

Epoch 7/30

- 0s - loss: 0.7993 - acc: 0.8270 - val\_loss: 0.9057 - val\_acc: 0.7530

- LR: 0.058824

Epoch 8/30

- 0s - loss: 0.7153 - acc: 0.8480 - val\_loss: 0.8433 - val\_acc: 0.7610

- LR: 0.055556

Epoch 9/30

- 0s - loss: 0.6547 - acc: 0.8530 - val\_loss: 0.8031 - val\_acc: 0.7790

- LR: 0.052632

Epoch 10/30

- 0s - loss: 0.6037 - acc: 0.8640 - val\_loss: 0.7634 - val\_acc: 0.7810

- LR: 0.050000

Epoch 11/30

- 0s - loss: 0.5657 - acc: 0.8730 - val\_loss: 0.7301 - val\_acc: 0.7930

- LR: 0.047619

Epoch 12/30

- 0s - loss: 0.5325 - acc: 0.8760 - val\_loss: 0.7075 - val\_acc: 0.8000

- LR: 0.045455

Epoch 13/30

- 0s - loss: 0.5047 - acc: 0.8840 - val\_loss: 0.6888 - val\_acc: 0.8000

- LR: 0.043478

Epoch 14/30

- 0s - loss: 0.4806 - acc: 0.8890 - val\_loss: 0.6733 - val\_acc: 0.8010

- LR: 0.041667

Epoch 15/30

- 0s - loss: 0.4597 - acc: 0.8970 - val\_loss: 0.6584 - val\_acc: 0.8110

- LR: 0.040000

Epoch 16/30

- 0s - loss: 0.4413 - acc: 0.8950 - val\_loss: 0.6463 - val\_acc: 0.8080

- LR: 0.038462

Epoch 17/30

- 0s - loss: 0.4262 - acc: 0.9060 - val\_loss: 0.6324 - val\_acc: 0.8130

- LR: 0.037037

Epoch 18/30

- 0s - loss: 0.4112 - acc: 0.9060 - val\_loss: 0.6235 - val\_acc: 0.8130

- LR: 0.035714

Epoch 19/30

- 0s - loss: 0.3987 - acc: 0.9130 - val\_loss: 0.6189 - val\_acc: 0.8110

- LR: 0.034483

Epoch 20/30

- 0s - loss: 0.3869 - acc: 0.9150 - val\_loss: 0.6067 - val\_acc: 0.8130

- LR: 0.033333

Epoch 21/30

- 0s - loss: 0.3769 - acc: 0.9170 - val\_loss: 0.5975 - val\_acc: 0.8210

- LR: 0.032258

Epoch 22/30

- 0s - loss: 0.3668 - acc: 0.9210 - val\_loss: 0.5924 - val\_acc: 0.8200

- LR: 0.031250

Epoch 23/30

- 0s - loss: 0.3574 - acc: 0.9270 - val\_loss: 0.5860 - val\_acc: 0.8270

- LR: 0.030303

Epoch 24/30

- 0s - loss: 0.3496 - acc: 0.9250 - val\_loss: 0.5794 - val\_acc: 0.8260

- LR: 0.029412

Epoch 25/30

- 0s - loss: 0.3418 - acc: 0.9280 - val\_loss: 0.5764 - val\_acc: 0.8270

- LR: 0.028571

Epoch 26/30

- 0s - loss: 0.3350 - acc: 0.9280 - val\_loss: 0.5727 - val\_acc: 0.8270

- LR: 0.027778

Epoch 27/30

- 0s - loss: 0.3283 - acc: 0.9290 - val\_loss: 0.5667 - val\_acc: 0.8290

- LR: 0.027027

Epoch 28/30

- 0s - loss: 0.3216 - acc: 0.9340 - val\_loss: 0.5628 - val\_acc: 0.8310

- LR: 0.026316

Epoch 29/30

- 0s - loss: 0.3162 - acc: 0.9350 - val\_loss: 0.5594 - val\_acc: 0.8280

- LR: 0.025641

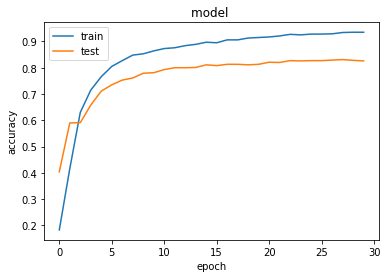
Epoch 30/30

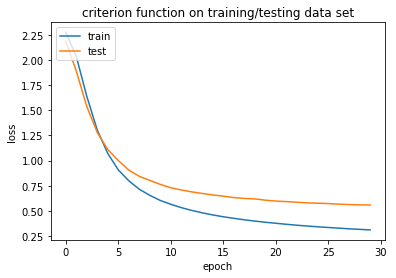
- 0s - loss: 0.3107 - acc: 0.9350 - val\_loss: 0.5588 - val\_acc: 0.8260

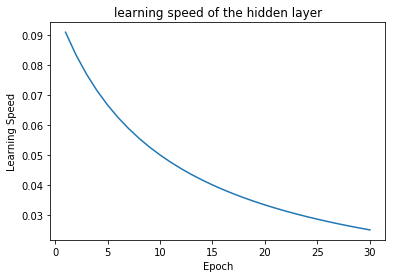
- LR: 0.025000

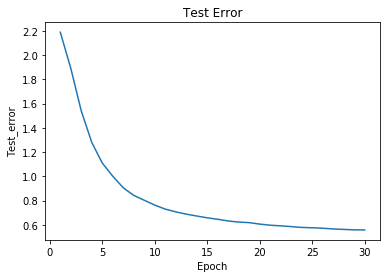
Baseline Error: 17.40%

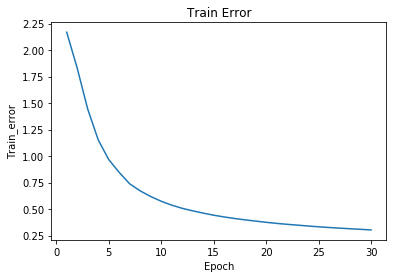
dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])











two\_hidden\_l2.py----two hidden layer with L2 regularization, lambda=5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_42 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_43 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_44 (Dense) (None, 10) 310

=================================================================

Total params: 24,790

Trainable params: 24,790

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

- 1s - loss: 2.3337 - acc: 0.0860 - val\_loss: 2.3111 - val\_acc: 0.1000

- LR: 0.099900

Epoch 2/30

- 0s - loss: 2.3334 - acc: 0.0860 - val\_loss: 2.3083 - val\_acc: 0.1000

- LR: 0.099800

Epoch 3/30

- 0s - loss: 2.3294 - acc: 0.0760 - val\_loss: 2.3134 - val\_acc: 0.1000

- LR: 0.099701

Epoch 4/30

- 0s - loss: 2.3223 - acc: 0.1050 - val\_loss: 2.3180 - val\_acc: 0.1000

- LR: 0.099602

Epoch 5/30

- 0s - loss: 2.3257 - acc: 0.0740 - val\_loss: 2.3209 - val\_acc: 0.1000

- LR: 0.099502

Epoch 6/30

- 0s - loss: 2.3181 - acc: 0.1010 - val\_loss: 2.3217 - val\_acc: 0.1000

- LR: 0.099404

Epoch 7/30

- 0s - loss: 2.3176 - acc: 0.1010 - val\_loss: 2.3045 - val\_acc: 0.2000

- LR: 0.099305

Epoch 8/30

- 0s - loss: 2.3046 - acc: 0.1190 - val\_loss: 2.3095 - val\_acc: 0.1000

- LR: 0.099206

Epoch 9/30

- 0s - loss: 2.2997 - acc: 0.1180 - val\_loss: 2.2889 - val\_acc: 0.1100

- LR: 0.099108

Epoch 10/30

- 0s - loss: 2.2789 - acc: 0.1570 - val\_loss: 2.2616 - val\_acc: 0.3160

- LR: 0.099010

Epoch 11/30

- 0s - loss: 2.2363 - acc: 0.2250 - val\_loss: 2.1993 - val\_acc: 0.2680

- LR: 0.098912

Epoch 12/30

- 0s - loss: 2.1403 - acc: 0.2780 - val\_loss: 2.1048 - val\_acc: 0.2720

- LR: 0.098814

Epoch 13/30

- 0s - loss: 2.0113 - acc: 0.2960 - val\_loss: 1.9582 - val\_acc: 0.3480

- LR: 0.098717

Epoch 14/30

- 0s - loss: 1.8519 - acc: 0.3470 - val\_loss: 1.8153 - val\_acc: 0.4220

- LR: 0.098619

Epoch 15/30

- 0s - loss: 1.6786 - acc: 0.4090 - val\_loss: 1.6510 - val\_acc: 0.4230

- LR: 0.098522

Epoch 16/30

- 0s - loss: 1.5159 - acc: 0.4600 - val\_loss: 1.5206 - val\_acc: 0.4700

- LR: 0.098425

Epoch 17/30

- 0s - loss: 1.3933 - acc: 0.5200 - val\_loss: 1.4278 - val\_acc: 0.5000

- LR: 0.098328

Epoch 18/30

- 0s - loss: 1.3023 - acc: 0.5550 - val\_loss: 1.3611 - val\_acc: 0.5510

- LR: 0.098232

Epoch 19/30

- 0s - loss: 1.2265 - acc: 0.5990 - val\_loss: 1.3087 - val\_acc: 0.5460

- LR: 0.098135

Epoch 20/30

- 0s - loss: 1.1578 - acc: 0.6180 - val\_loss: 1.2706 - val\_acc: 0.5480

- LR: 0.098039

Epoch 21/30

- 0s - loss: 1.0983 - acc: 0.6640 - val\_loss: 1.2153 - val\_acc: 0.6070

- LR: 0.097943

Epoch 22/30

- 0s - loss: 1.0413 - acc: 0.6840 - val\_loss: 1.1766 - val\_acc: 0.6120

- LR: 0.097847

Epoch 23/30

- 0s - loss: 0.9885 - acc: 0.7230 - val\_loss: 1.1203 - val\_acc: 0.6540

- LR: 0.097752

Epoch 24/30

- 0s - loss: 0.9293 - acc: 0.7460 - val\_loss: 1.1028 - val\_acc: 0.6380

- LR: 0.097656

Epoch 25/30

- 0s - loss: 0.8762 - acc: 0.7600 - val\_loss: 1.0502 - val\_acc: 0.6750

- LR: 0.097561

Epoch 26/30

- 0s - loss: 0.8294 - acc: 0.7700 - val\_loss: 1.0168 - val\_acc: 0.6910

- LR: 0.097466

Epoch 27/30

- 0s - loss: 0.7853 - acc: 0.7970 - val\_loss: 0.9851 - val\_acc: 0.7060

- LR: 0.097371

Epoch 28/30

- 0s - loss: 0.7427 - acc: 0.8100 - val\_loss: 0.9552 - val\_acc: 0.7090

- LR: 0.097276

Epoch 29/30

- 0s - loss: 0.7059 - acc: 0.8140 - val\_loss: 0.9366 - val\_acc: 0.7140

- LR: 0.097182

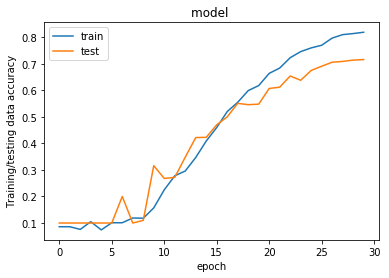
Epoch 30/30

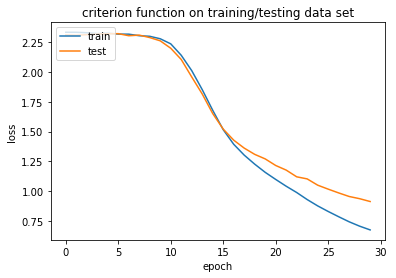
- 0s - loss: 0.6746 - acc: 0.8190 - val\_loss: 0.9129 - val\_acc: 0.7160

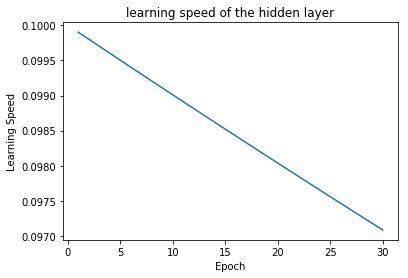
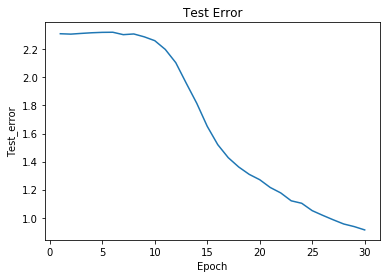
- LR: 0.097087

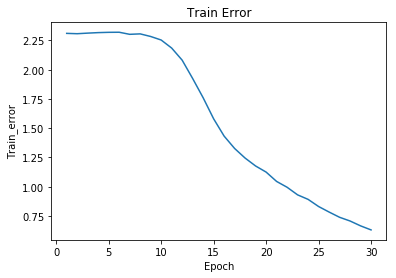
Baseline Error: 28.40%

dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])









output of three\_hidden\_l2.py ---- three hidden layer with L2 regularization, lambda=5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_46 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_47 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_48 (Dense) (None, 30) 930

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_49 (Dense) (None, 10) 310

=================================================================

Total params: 25,720

Trainable params: 25,720

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

- 1s - loss: 2.3351 - acc: 0.0830 - val\_loss: 2.3130 - val\_acc: 0.1000

- LR: 0.099010

Epoch 2/30

- 0s - loss: 2.3376 - acc: 0.0820 - val\_loss: 2.3154 - val\_acc: 0.1000

- LR: 0.098039

Epoch 3/30

- 0s - loss: 2.3357 - acc: 0.0790 - val\_loss: 2.3136 - val\_acc: 0.1000

- LR: 0.097087

Epoch 4/30

- 0s - loss: 2.3282 - acc: 0.1010 - val\_loss: 2.3226 - val\_acc: 0.1000

- LR: 0.096154

Epoch 5/30

- 0s - loss: 2.3324 - acc: 0.0940 - val\_loss: 2.3248 - val\_acc: 0.1000

- LR: 0.095238

Epoch 6/30

- 0s - loss: 2.3249 - acc: 0.0990 - val\_loss: 2.3303 - val\_acc: 0.1000

- LR: 0.094340

Epoch 7/30

- 0s - loss: 2.3307 - acc: 0.0990 - val\_loss: 2.3114 - val\_acc: 0.1000

- LR: 0.093458

Epoch 8/30

- 0s - loss: 2.3277 - acc: 0.0800 - val\_loss: 2.3146 - val\_acc: 0.1000

- LR: 0.092593

Epoch 9/30

- 0s - loss: 2.3229 - acc: 0.0880 - val\_loss: 2.3205 - val\_acc: 0.1000

- LR: 0.091743

Epoch 10/30

- 0s - loss: 2.3272 - acc: 0.0820 - val\_loss: 2.3128 - val\_acc: 0.1000

- LR: 0.090909

Epoch 11/30

- 0s - loss: 2.3247 - acc: 0.0780 - val\_loss: 2.3104 - val\_acc: 0.1000

- LR: 0.090090

Epoch 12/30

- 0s - loss: 2.3230 - acc: 0.1000 - val\_loss: 2.3212 - val\_acc: 0.1000

- LR: 0.089286

Epoch 13/30

- 0s - loss: 2.3268 - acc: 0.0810 - val\_loss: 2.3152 - val\_acc: 0.1000

- LR: 0.088496

Epoch 14/30

- 0s - loss: 2.3223 - acc: 0.0930 - val\_loss: 2.3130 - val\_acc: 0.1000

- LR: 0.087719

Epoch 15/30

- 0s - loss: 2.3205 - acc: 0.0910 - val\_loss: 2.3186 - val\_acc: 0.1000

- LR: 0.086957

Epoch 16/30

- 0s - loss: 2.3255 - acc: 0.0880 - val\_loss: 2.3134 - val\_acc: 0.1000

- LR: 0.086207

Epoch 17/30

- 0s - loss: 2.3211 - acc: 0.0960 - val\_loss: 2.3135 - val\_acc: 0.1000

- LR: 0.085470

Epoch 18/30

- 0s - loss: 2.3220 - acc: 0.0840 - val\_loss: 2.3113 - val\_acc: 0.1000

- LR: 0.084746

Epoch 19/30

- 0s - loss: 2.3224 - acc: 0.0850 - val\_loss: 2.3125 - val\_acc: 0.1000

- LR: 0.084034

Epoch 20/30

- 0s - loss: 2.3218 - acc: 0.0860 - val\_loss: 2.3126 - val\_acc: 0.1000

- LR: 0.083333

Epoch 21/30

- 0s - loss: 2.3226 - acc: 0.0780 - val\_loss: 2.3107 - val\_acc: 0.1000

- LR: 0.082645

Epoch 22/30

- 0s - loss: 2.3174 - acc: 0.1070 - val\_loss: 2.3152 - val\_acc: 0.1000

- LR: 0.081967

Epoch 23/30

- 0s - loss: 2.3196 - acc: 0.1080 - val\_loss: 2.3120 - val\_acc: 0.1000

- LR: 0.081301

Epoch 24/30

- 0s - loss: 2.3153 - acc: 0.0970 - val\_loss: 2.3198 - val\_acc: 0.1000

- LR: 0.080645

Epoch 25/30

- 0s - loss: 2.3225 - acc: 0.0930 - val\_loss: 2.3123 - val\_acc: 0.1000

- LR: 0.080000

Epoch 26/30

- 0s - loss: 2.3181 - acc: 0.1140 - val\_loss: 2.3118 - val\_acc: 0.1000

- LR: 0.079365

Epoch 27/30

- 0s - loss: 2.3184 - acc: 0.0880 - val\_loss: 2.3116 - val\_acc: 0.1000

- LR: 0.078740

Epoch 28/30

- 0s - loss: 2.3185 - acc: 0.0920 - val\_loss: 2.3106 - val\_acc: 0.1000

- LR: 0.078125

Epoch 29/30

- 0s - loss: 2.3185 - acc: 0.0900 - val\_loss: 2.3110 - val\_acc: 0.1000

- LR: 0.077519

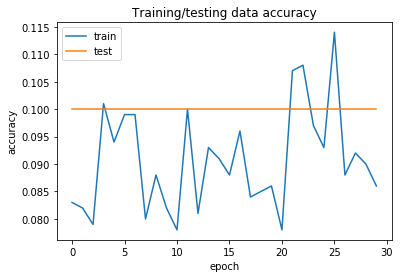
Epoch 30/30

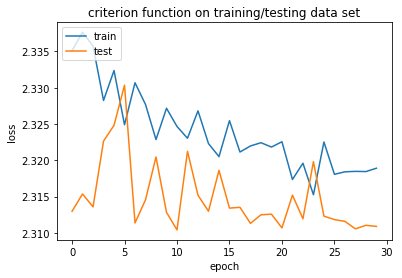
- 0s - loss: 2.3189 - acc: 0.0860 - val\_loss: 2.3109 - val\_acc: 0.1000

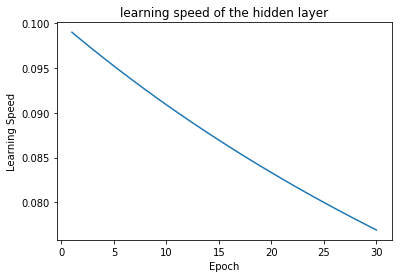
- LR: 0.076923

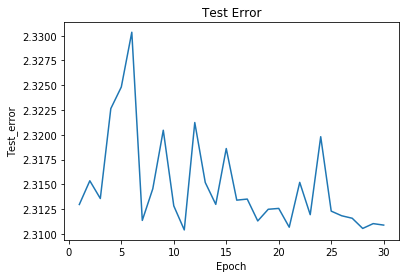
Baseline Error: 10.00%

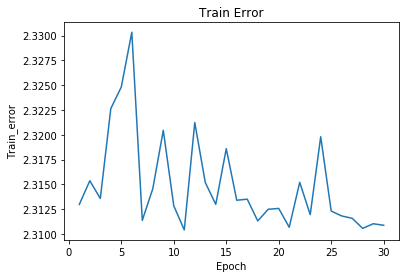
dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])











2 (c). (5 points) Construct and train convolutional neural network for MNIST classification. Regularize the training of the neural network through dropout. Regularize the training of neural network through augment your selection of 1000 images by rotating them for 1-3 degrees clockwise and counter clockwise, and shifting them for 3 pixels in 8 different directions. You can find many tutorials on those techniques, and our emphasize is that we understand those techniques.

Soln:

Construct and train convolutional neural network for MNIST classification.

This is without dropout.

Filename: **MNIST\_convo.py**

This was run on the whole 60000 training data and the 10000 testing data

It was run for 10 epochs. The output is below:

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

- 171s - loss: 0.2331 - acc: 0.9334 - val\_loss: 0.0800 - val\_acc: 0.9756

Epoch 2/10

- 163s - loss: 0.0661 - acc: 0.9805 - val\_loss: 0.0640 - val\_acc: 0.9785

Epoch 3/10

- 162s - loss: 0.0452 - acc: 0.9864 - val\_loss: 0.0422 - val\_acc: 0.9856

Epoch 4/10

- 160s - loss: 0.0338 - acc: 0.9895 - val\_loss: 0.0395 - val\_acc: 0.9859

Epoch 5/10

- 163s - loss: 0.0273 - acc: 0.9913 - val\_loss: 0.0385 - val\_acc: 0.9869

Epoch 6/10

- 185s - loss: 0.0213 - acc: 0.9936 - val\_loss: 0.0376 - val\_acc: 0.9875

Epoch 7/10

- 234s - loss: 0.0163 - acc: 0.9954 - val\_loss: 0.0396 - val\_acc: 0.9874

Epoch 8/10

- 214s - loss: 0.0124 - acc: 0.9962 - val\_loss: 0.0377 - val\_acc: 0.9887

Epoch 9/10

- 212s - loss: 0.0097 - acc: 0.9972 - val\_loss: 0.0359 - val\_acc: 0.9887

Epoch 10/10

- 176s - loss: 0.0088 - acc: 0.9973 - val\_loss: 0.0369 - val\_acc: 0.9887

CNN Error: 1.13%

THE accuracy is 98.87 percent and the error is 1.13%.

With Dropout of 20 percent: File name is ***MNIST\_convo\_dropout.py***

This was run on the whole 60000 training data and the 10000 testing data

It was run for 10 epochs. The output is below:

Train on 60000 samples, validate on 10000 samples

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

- 223s - loss: 0.2315 - acc: 0.9343 - val\_loss: 0.0815 - val\_acc: 0.9743

Epoch 2/10

- 198s - loss: 0.0738 - acc: 0.9781 - val\_loss: 0.0469 - val\_acc: 0.9839

Epoch 3/10

- 178s - loss: 0.0532 - acc: 0.9839 - val\_loss: 0.0425 - val\_acc: 0.9862

Epoch 4/10

- 180s - loss: 0.0403 - acc: 0.9879 - val\_loss: 0.0402 - val\_acc: 0.9869

Epoch 5/10

- 186s - loss: 0.0336 - acc: 0.9894 - val\_loss: 0.0341 - val\_acc: 0.9883

Epoch 6/10

- 185s - loss: 0.0273 - acc: 0.9915 - val\_loss: 0.0301 - val\_acc: 0.9899

Epoch 7/10

- 171s - loss: 0.0233 - acc: 0.9927 - val\_loss: 0.0342 - val\_acc: 0.9886

Epoch 8/10

- 166s - loss: 0.0202 - acc: 0.9938 - val\_loss: 0.0324 - val\_acc: 0.9882

Epoch 9/10

- 167s - loss: 0.0169 - acc: 0.9944 - val\_loss: 0.0297 - val\_acc: 0.9901

Epoch 10/10

- 164s - loss: 0.0142 - acc: 0.9960 - val\_loss: 0.0316 - val\_acc: 0.9910

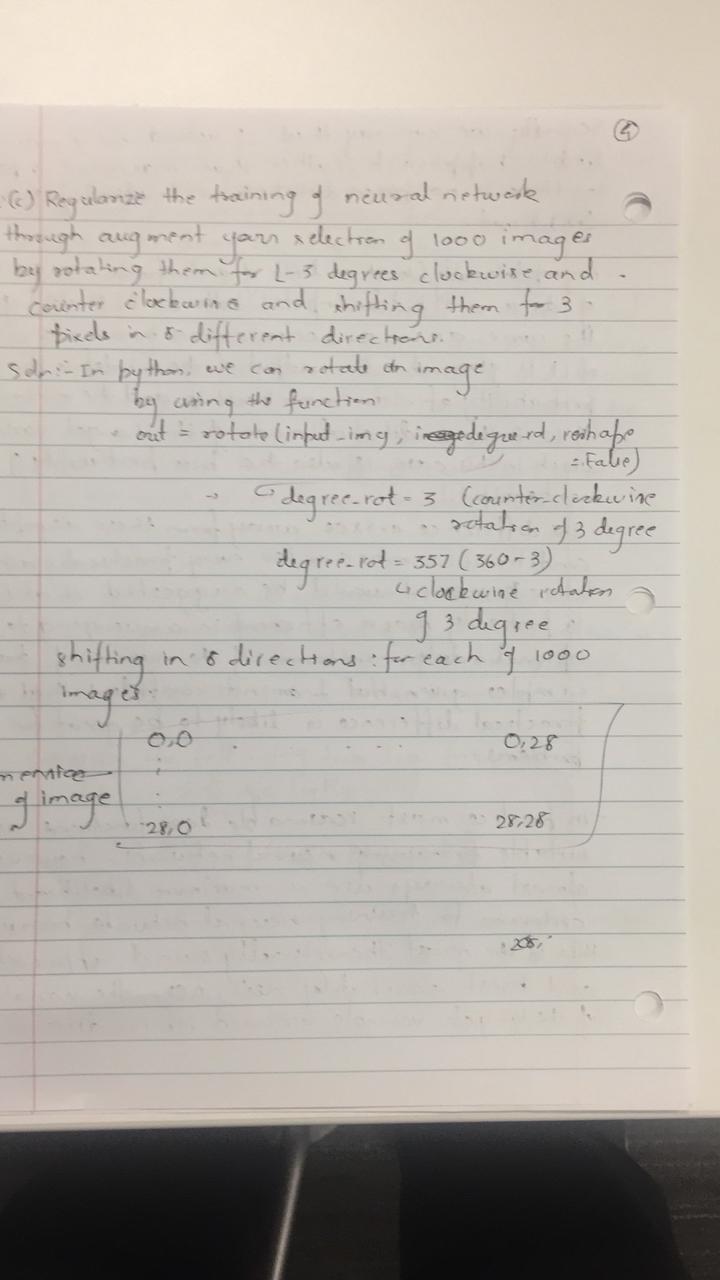
CNN Error: 0.90%

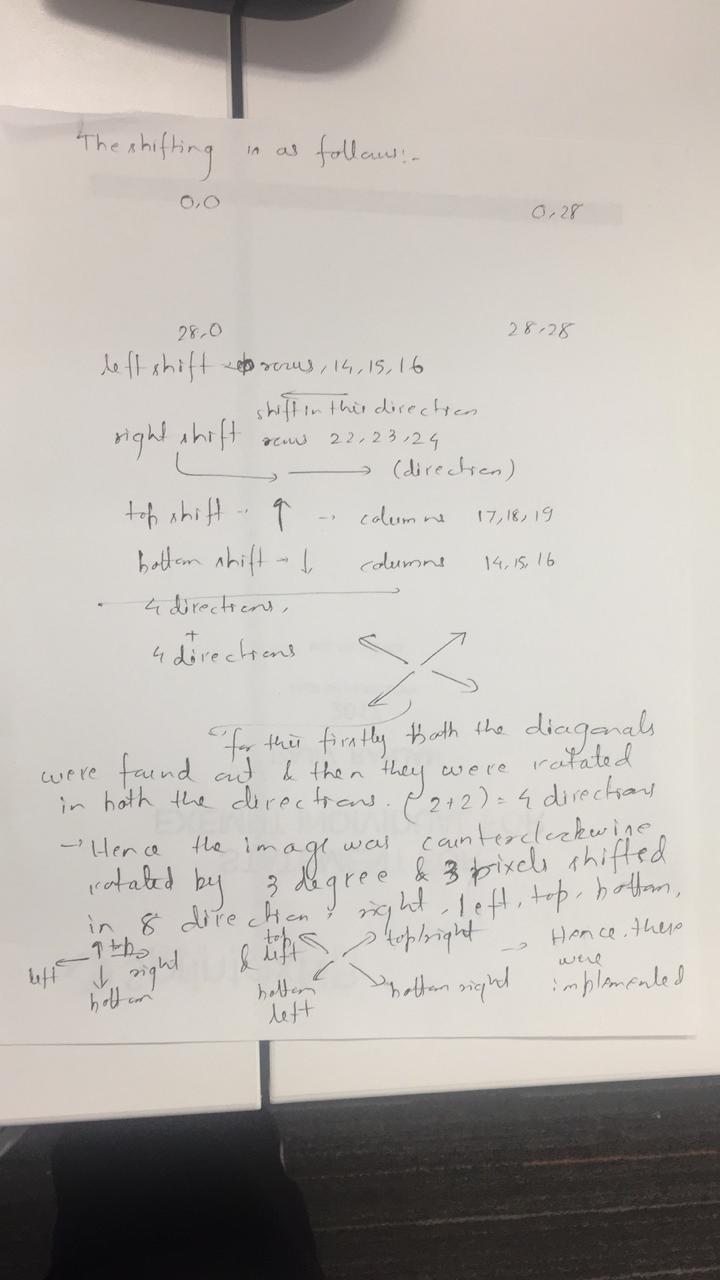
**As we see that with a dropout of 20 percent the accuracy increases:**

**Now the accuracy is 99.10 percent and the error is just 0.90 percent.**

c) Regularize the training of neural network through augment your selection of 1000 images by rotating them for 1-3 degrees clockwise and counter clockwise, and shifting them for 3 pixels in 8 different directions. You can find many tutorials on those techniques, and our emphasize is that we understand those techniques.

Soln: The output is in file:**rotated\_neural.py**

****



The output of the execution is below:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_58 (Dense) (None, 30) 23550

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_59 (Dense) (None, 10) 310

=================================================================

Total params: 23,860

Trainable params: 23,860

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train on 1000 samples, validate on 1000 samples

Epoch 1/30

/Users/rajivranjan/anaconda3/lib/python3.6/site-packages/keras/models.py:942: UserWarning: The `nb\_epoch` argument in `fit` has been renamed `epochs`.

warnings.warn('The `nb\_epoch` argument in `fit` '

- 1s - loss: 2.2702 - acc: 0.1840 - val\_loss: 2.1824 - val\_acc: 0.4030

- LR: 0.090909

Epoch 2/30

- 0s - loss: 2.0296 - acc: 0.4150 - val\_loss: 1.8840 - val\_acc: 0.5890

- LR: 0.083333

Epoch 3/30

- 0s - loss: 1.6351 - acc: 0.6260 - val\_loss: 1.5323 - val\_acc: 0.5910

- LR: 0.076923

Epoch 4/30

- 0s - loss: 1.2986 - acc: 0.7150 - val\_loss: 1.2722 - val\_acc: 0.6600

- LR: 0.071429

Epoch 5/30

- 0s - loss: 1.0656 - acc: 0.7660 - val\_loss: 1.1014 - val\_acc: 0.7120

- LR: 0.066667

Epoch 6/30

- 0s - loss: 0.9016 - acc: 0.8070 - val\_loss: 0.9920 - val\_acc: 0.7340

- LR: 0.062500

Epoch 7/30

- 0s - loss: 0.7928 - acc: 0.8260 - val\_loss: 0.8961 - val\_acc: 0.7540

- LR: 0.058824

Epoch 8/30

- 0s - loss: 0.7084 - acc: 0.8480 - val\_loss: 0.8332 - val\_acc: 0.7620

- LR: 0.055556

Epoch 9/30

- 0s - loss: 0.6473 - acc: 0.8520 - val\_loss: 0.7925 - val\_acc: 0.7800

- LR: 0.052632

Epoch 10/30

- 0s - loss: 0.5960 - acc: 0.8650 - val\_loss: 0.7525 - val\_acc: 0.7820

- LR: 0.050000

Epoch 11/30

- 0s - loss: 0.5576 - acc: 0.8720 - val\_loss: 0.7189 - val\_acc: 0.7960

- LR: 0.047619

Epoch 12/30

- 0s - loss: 0.5241 - acc: 0.8750 - val\_loss: 0.6959 - val\_acc: 0.8020

- LR: 0.045455

Epoch 13/30

- 0s - loss: 0.4960 - acc: 0.8830 - val\_loss: 0.6770 - val\_acc: 0.7990

- LR: 0.043478

Epoch 14/30

- 0s - loss: 0.4717 - acc: 0.8890 - val\_loss: 0.6613 - val\_acc: 0.8000

- LR: 0.041667

Epoch 15/30

- 0s - loss: 0.4506 - acc: 0.8960 - val\_loss: 0.6461 - val\_acc: 0.8100

- LR: 0.040000

Epoch 16/30

- 0s - loss: 0.4319 - acc: 0.8950 - val\_loss: 0.6338 - val\_acc: 0.8090

- LR: 0.038462

Epoch 17/30

- 0s - loss: 0.4166 - acc: 0.9040 - val\_loss: 0.6197 - val\_acc: 0.8110

- LR: 0.037037

Epoch 18/30

- 0s - loss: 0.4014 - acc: 0.9050 - val\_loss: 0.6106 - val\_acc: 0.8120

- LR: 0.035714

Epoch 19/30

- 0s - loss: 0.3887 - acc: 0.9140 - val\_loss: 0.6057 - val\_acc: 0.8120

- LR: 0.034483

Epoch 20/30

- 0s - loss: 0.3768 - acc: 0.9140 - val\_loss: 0.5936 - val\_acc: 0.8150

- LR: 0.033333

Epoch 21/30

- 0s - loss: 0.3665 - acc: 0.9160 - val\_loss: 0.5843 - val\_acc: 0.8230

- LR: 0.032258

Epoch 22/30

- 0s - loss: 0.3563 - acc: 0.9200 - val\_loss: 0.5790 - val\_acc: 0.8220

- LR: 0.031250

Epoch 23/30

- 0s - loss: 0.3467 - acc: 0.9250 - val\_loss: 0.5726 - val\_acc: 0.8290

- LR: 0.030303

Epoch 24/30

- 0s - loss: 0.3388 - acc: 0.9240 - val\_loss: 0.5658 - val\_acc: 0.8290

- LR: 0.029412

Epoch 25/30

- 0s - loss: 0.3309 - acc: 0.9270 - val\_loss: 0.5629 - val\_acc: 0.8270

- LR: 0.028571

Epoch 26/30

- 0s - loss: 0.3240 - acc: 0.9280 - val\_loss: 0.5590 - val\_acc: 0.8270

- LR: 0.027778

Epoch 27/30

- 0s - loss: 0.3172 - acc: 0.9280 - val\_loss: 0.5529 - val\_acc: 0.8290

- LR: 0.027027

Epoch 28/30

- 0s - loss: 0.3104 - acc: 0.9330 - val\_loss: 0.5489 - val\_acc: 0.8310

- LR: 0.026316

Epoch 29/30

- 0s - loss: 0.3048 - acc: 0.9340 - val\_loss: 0.5454 - val\_acc: 0.8300

- LR: 0.025641

Epoch 30/30

- 0s - loss: 0.2992 - acc: 0.9340 - val\_loss: 0.5447 - val\_acc: 0.8260

- LR: 0.025000

Baseline Error: 17.40%

dict\_keys(['val\_loss', 'val\_acc', 'loss', 'acc'])

The accuracy is 82.60 percent and loss is 17.40 percent.

3. (Optional) Train GAN to generate the images for the 10 digits from random noise. Train autoencoder network with linear and sigmoid activation functions for principle component analysis. Train recurrent neural network to accept the 28 rows and output the digit of the image.

Soln:

1. Train GAN to generate the images for the 10 digits from random noise.

Soln: Filename: GANquestion.py

Here “z” is used is the random noise generator

Output is below:

It runs for 10 epochs:

Extracting ../MNIST\_data/train-images-idx3-ubyte.gz

Extracting ../MNIST\_data/train-labels-idx1-ubyte.gz

Extracting ../MNIST\_data/t10k-images-idx3-ubyte.gz

Extracting ../MNIST\_data/t10k-labels-idx1-ubyte.gz

['dis/dense/kernel:0', 'dis/dense/bias:0', 'dis/dense\_1/kernel:0', 'dis/dense\_1/bias:0', 'dis/dense\_2/kernel:0', 'dis/dense\_2/bias:0']

['gen/dense/kernel:0', 'gen/dense/bias:0', 'gen/dense\_1/kernel:0', 'gen/dense\_1/bias:0', 'gen/dense\_2/kernel:0', 'gen/dense\_2/bias:0']

Currently on Epoch 1 of 10 total...

Currently on Epoch 2 of 10 total...

Currently on Epoch 3 of 10 total...

Currently on Epoch 4 of 10 total...

Currently on Epoch 5 of 10 total...

Currently on Epoch 6 of 10 total...

Currently on Epoch 7 of 10 total...

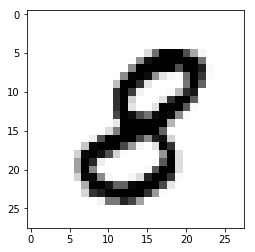
Currently on Epoch 8 of 10 total...

Currently on Epoch 9 of 10 total...

Currently on Epoch 10 of 10 total...

INFO:tensorflow:Restoring parameters from ./models/500\_epoch\_model.ckpt

We see one of the samples created as follows:It’s a digit 8.



1. Train autoencoder network with linear and sigmoid activation functions for principle component analysis.

Soln:

Linear\_autoencoder:filename is autoencoder\_linear.py

The output is below:

It runs for 10 epochs:

Using TensorFlow backend.

(60000, 784)

(10000, 784)

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [==============================] - 7s 115us/step - loss: 0.3381 - val\_loss: 0.2420

Epoch 2/10

60000/60000 [==============================] - 6s 101us/step - loss: 0.2178 - val\_loss: 0.1997

Epoch 3/10

60000/60000 [==============================] - 7s 110us/step - loss: 0.1877 - val\_loss: 0.1762

Epoch 4/10

60000/60000 [==============================] - 7s 112us/step - loss: 0.1727 - val\_loss: 0.1659

Epoch 5/10

60000/60000 [==============================] - 7s 109us/step - loss: 0.1630 - val\_loss: 0.1574

Epoch 6/10

60000/60000 [==============================] - 7s 111us/step - loss: 0.1519 - val\_loss: 0.1468

Epoch 7/10

60000/60000 [==============================] - 7s 111us/step - loss: 0.1457 - val\_loss: 0.1426

Epoch 8/10

60000/60000 [==============================] - 7s 112us/step - loss: 0.1416 - val\_loss: 0.1375

Epoch 9/10

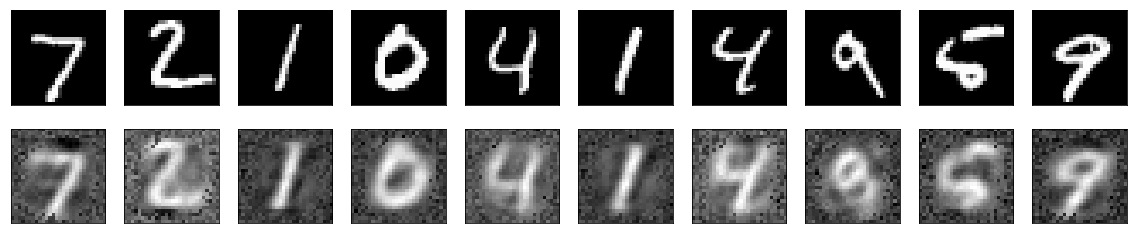
60000/60000 [==============================] - 7s 117us/step - loss: 0.1383 - val\_loss: 0.1347

Epoch 10/10

60000/60000 [==============================] - 6s 108us/step - loss: 0.1358 - val\_loss: 0.1328

The accuracy is 88 percent and the loss is 12 percent.

Below is the output of the linear encoder used for PCA.



Sigmoid\_autoencoder:filename is autoencoder\_sigmoid.py

The output is below:

It runs for 10 epochs:

(60000, 784)

(10000, 784)

Train on 60000 samples, validate on 10000 samples

Epoch 1/10

60000/60000 [==============================] - 7s 115us/step - loss: 0.3627 - val\_loss: 0.2712

Epoch 2/10

60000/60000 [==============================] - 7s 111us/step - loss: 0.2637 - val\_loss: 0.2524

Epoch 3/10

60000/60000 [==============================] - 6s 103us/step - loss: 0.2414 - val\_loss: 0.2284

Epoch 4/10

60000/60000 [==============================] - 6s 104us/step - loss: 0.2210 - val\_loss: 0.2113

Epoch 5/10

60000/60000 [==============================] - 6s 108us/step - loss: 0.2067 - val\_loss: 0.1995

Epoch 6/10

60000/60000 [==============================] - 7s 108us/step - loss: 0.1961 - val\_loss: 0.1900

Epoch 7/10

60000/60000 [==============================] - 7s 115us/step - loss: 0.1875 - val\_loss: 0.1821

Epoch 8/10

60000/60000 [==============================] - 6s 108us/step - loss: 0.1803 - val\_loss: 0.1757

Epoch 9/10

60000/60000 [==============================] - 6s 104us/step - loss: 0.1743 - val\_loss: 0.1701

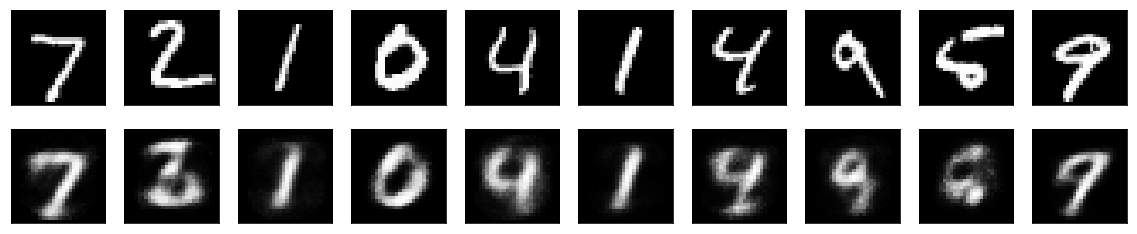
Epoch 10/10

60000/60000 [==============================] - 7s 120us/step - loss: 0.1691 - val\_loss: 0.1652

The accuracy is 84 percent and the loss is 16 percent.

Below is the output of the linear encoder used for PCA.

A sample of 10 digits is shown below:



Inference:

Sigmoid performs much better than the linear autoencoder as we can see clearly from the outputs generated by the execution of both the two types of autoencoders.

Also dimensionality reduction was done as follows:

encoding\_dim = 32

# 32 floats -> compression of factor 24.5, assuming the input is 784 floats

c)Train recurrent neural network to accept the 28 rows and output the digit of the image.

Soln: The output is in file **Rnn\_ques.py**

**The output is below:**

**It was run for 800 iterations with a batch size of 128.**

**The accuracy is 98.43 percent and 1.5 percent is the error rate.**

**The o/p is:**

Extracting /tmp/data/train-images-idx3-ubyte.gz

Extracting /tmp/data/train-labels-idx1-ubyte.gz

Extracting /tmp/data/t10k-images-idx3-ubyte.gz

Extracting /tmp/data/t10k-labels-idx1-ubyte.gz

WARNING:tensorflow:From /Users/rajivranjan/Desktop/Rnn\_ques.py:45: softmax\_cross\_entropy\_with\_logits (from tensorflow.python.ops.nn\_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow

into the labels input on backprop by default.

See tf.nn.softmax\_cross\_entropy\_with\_logits\_v2.

For iter 10

Accuracy 0.320312

Loss 1.97964

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 20

Accuracy 0.59375

Loss 1.3016

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 30

Accuracy 0.609375

Loss 1.18073

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 40

Accuracy 0.648438

Loss 1.06685

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 50

Accuracy 0.695312

Loss 0.84303

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 60

Accuracy 0.703125

Loss 0.959818

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 70

Accuracy 0.789062

Loss 0.665865

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 80

Accuracy 0.851562

Loss 0.454843

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 90

Accuracy 0.789062

Loss 0.513934

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 100

Accuracy 0.867188

Loss 0.63647

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 110

Accuracy 0.820312

Loss 0.505687

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 120

Accuracy 0.867188

Loss 0.467315

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 130

Accuracy 0.914062

Loss 0.286141

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 140

Accuracy 0.882812

Loss 0.401626

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 150

Accuracy 0.90625

Loss 0.358282

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 160

Accuracy 0.882812

Loss 0.317586

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 170

Accuracy 0.921875

Loss 0.2495

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 180

Accuracy 0.875

Loss 0.351491

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 190

Accuracy 0.914062

Loss 0.264019

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 200

Accuracy 0.921875

Loss 0.238182

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 210

Accuracy 0.914062

Loss 0.264488

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 220

Accuracy 0.960938

Loss 0.185066

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 230

Accuracy 0.898438

Loss 0.25824

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 240

Accuracy 0.9375

Loss 0.223796

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 250

Accuracy 0.929688

Loss 0.284195

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 260

Accuracy 0.929688

Loss 0.254908

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 270

Accuracy 0.921875

Loss 0.20947

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 280

Accuracy 0.914062

Loss 0.241348

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 290

Accuracy 0.945312

Loss 0.190349

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 300

Accuracy 0.9375

Loss 0.198062

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 310

Accuracy 0.921875

Loss 0.213394

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 320

Accuracy 0.921875

Loss 0.228114

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 330

Accuracy 0.914062

Loss 0.150239

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 340

Accuracy 0.960938

Loss 0.132705

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 350

Accuracy 0.914062

Loss 0.310001

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 360

Accuracy 0.953125

Loss 0.127136

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 370

Accuracy 0.921875

Loss 0.27158

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 380

Accuracy 0.960938

Loss 0.130141

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 390

Accuracy 0.953125

Loss 0.167404

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 400

Accuracy 0.914062

Loss 0.22073

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 410

Accuracy 0.929688

Loss 0.180388

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 420

Accuracy 0.945312

Loss 0.181368

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 430

Accuracy 0.96875

Loss 0.123066

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 440

Accuracy 0.96875

Loss 0.0826505

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 450

Accuracy 0.945312

Loss 0.14263

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 460

Accuracy 0.984375

Loss 0.0508476

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 470

Accuracy 0.984375

Loss 0.0804394

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 480

Accuracy 0.976562

Loss 0.129626

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 490

Accuracy 0.945312

Loss 0.180093

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 500

Accuracy 0.945312

Loss 0.160532

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 510

Accuracy 0.992188

Loss 0.0616836

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 520

Accuracy 0.960938

Loss 0.130104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 530

Accuracy 0.96875

Loss 0.110087

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 540

Accuracy 0.960938

Loss 0.087577

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 550

Accuracy 0.992188

Loss 0.0458895

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 560

Accuracy 0.9375

Loss 0.169928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 570

Accuracy 0.976562

Loss 0.16944

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 580

Accuracy 0.929688

Loss 0.203101

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 590

Accuracy 0.960938

Loss 0.130237

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 600

Accuracy 0.929688

Loss 0.16592

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 610

Accuracy 0.96875

Loss 0.157914

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 620

Accuracy 0.945312

Loss 0.174224

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 630

Accuracy 0.953125

Loss 0.13913

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 640

Accuracy 0.984375

Loss 0.0620078

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 650

Accuracy 0.976562

Loss 0.0557024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 660

Accuracy 0.96875

Loss 0.13908

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 670

Accuracy 0.984375

Loss 0.0445744

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 680

Accuracy 0.953125

Loss 0.22088

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 690

Accuracy 0.960938

Loss 0.118377

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 700

Accuracy 0.921875

Loss 0.169052

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 710

Accuracy 0.945312

Loss 0.155573

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 720

Accuracy 0.960938

Loss 0.0951964

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 730

Accuracy 0.96875

Loss 0.104905

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 740

Accuracy 0.984375

Loss 0.0558364

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 750

Accuracy 0.984375

Loss 0.0479834

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 760

Accuracy 0.960938

Loss 0.116536

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 770

Accuracy 0.96875

Loss 0.111931

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 780

Accuracy 0.960938

Loss 0.149431

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For iter 790

Accuracy 0.960938

Loss 0.129065

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Testing Accuracy: 0.984375

4. (Optional) Train Bayesian neural network with variational and sampling based method using [Edward](http://edwardlib.org/) and Tensorflow. We will cover Bayesian neural network in the lecture.

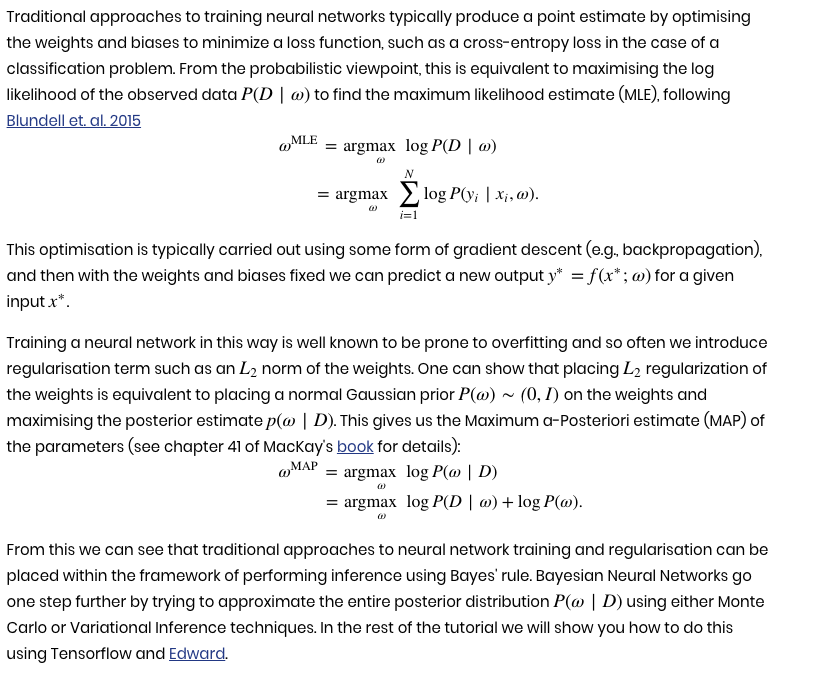
Soln:

The filename is: variational\_sampling.py

The output is below: The code runs for around 3 hours.

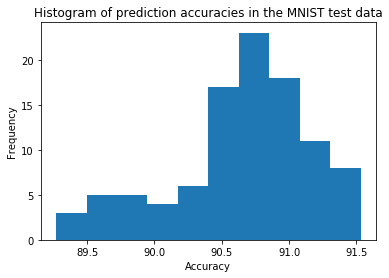
The code also runs only for the Tensorflow version 1.2.0 .

I have constructed a simple Bayesian statistical model for MNIST image classification using TensorFlow and Edward. Understanding uncertainty in statistical inference is very important for a variety of applications and we have explored some basic methods for visualising this problem.



The output is below:





We should also look at the posterior distribution. Unfortunately, the number of dimensions is quite large even for a small problem like this and so visualising them is tricky! We look at the first 5 dimensions and produce a triangle plot of the correlations.

