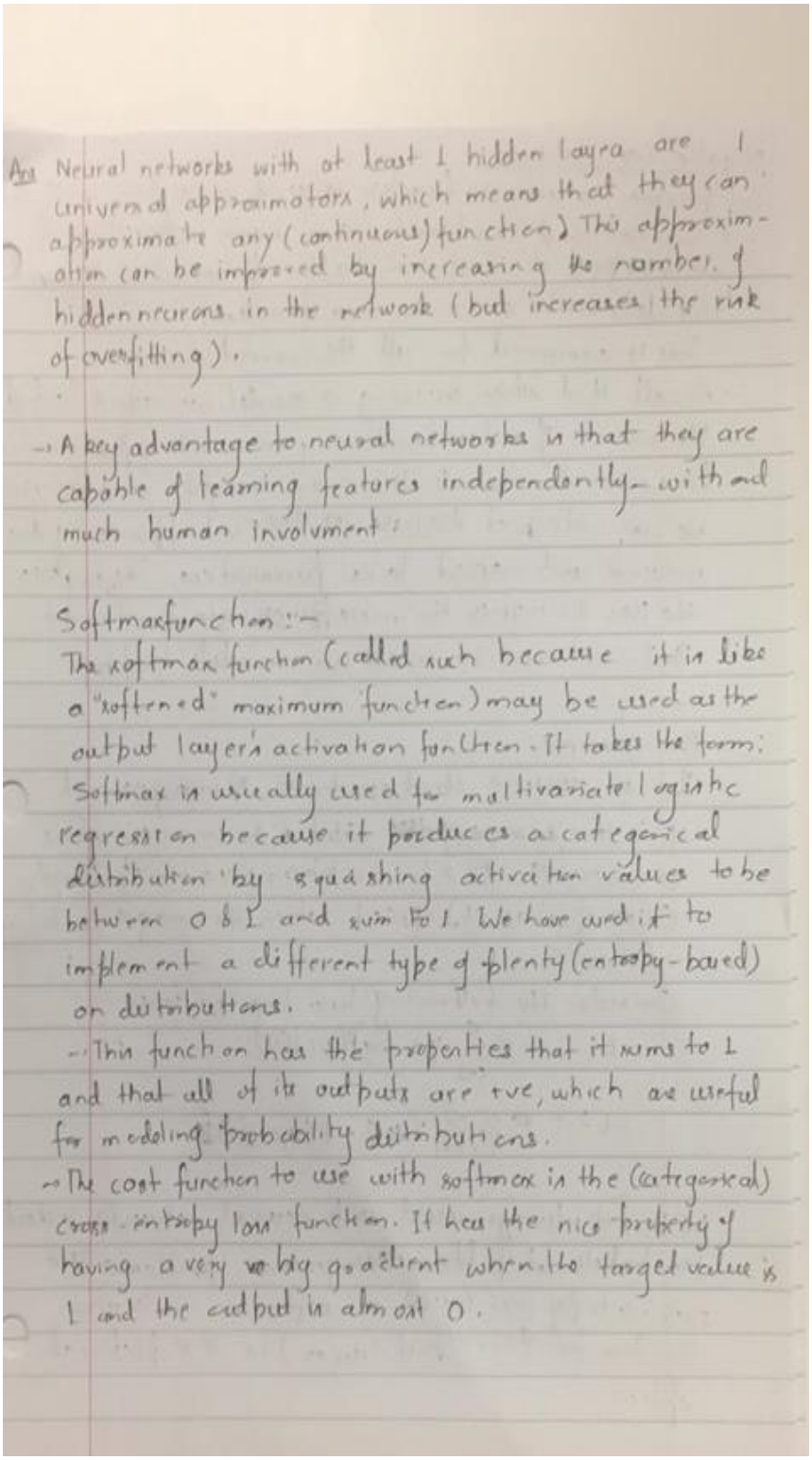
PROBLEM SET 4 – NEURAL NETWORKS

Ravi Teja Sunkara (rsunkara) (50292191)

UNIVERSITY AT BUFFALO (rsunkara@buffalo.edu)

Question 1



Ans Neural networks with at least 1 hidden layer are 1 universal approximators, which means that they can approximate any (continuous) function. This approximation can be improved by increasing the number of hidden neurons in the network (but increases the risk of overfitting).

→ A key advantage to neural networks is that they are capable of learning features independently with and much human involvement.

Softmax function :-

The softmax function (called such because it is like a "softened" maximum function) may be used as the output layer's activation function. It takes the form:

Softmax is usually used for multivariate logistic regression because it produces a categorical distribution by squashing activation values to be between 0 & 1 and sum to 1. We have used it to implement a different type of plenty (entropy-based) on distributions.

→ This function has the properties that it sums to 1 and that all of its outputs are +ve, which are useful for modeling probability distributions.

→ The cost function to use with softmax is the (categorical) cross-entropy loss function. It has the nice property of having a very big gradient when the target value is 1 and the output is almost 0.

Negative Log-Likelihood :-

In practice, the softmax function is used in tandem with the -ve log-likelihood. This loss function is very interesting if we interpret it in relation to the behavior of softmax.

First, let's write down our loss function:

This is summed for all the correct classes.

- Recall that when training a model, we aspire to find the minima of a loss function given a set of parameters (in a neural network, these are weights and biases). We can interpret the loss as the "unhappiness" of the network with respect to its parameters. The higher the loss, the higher the unhappiness: we don't want that. We want to make our models happy.

For example:

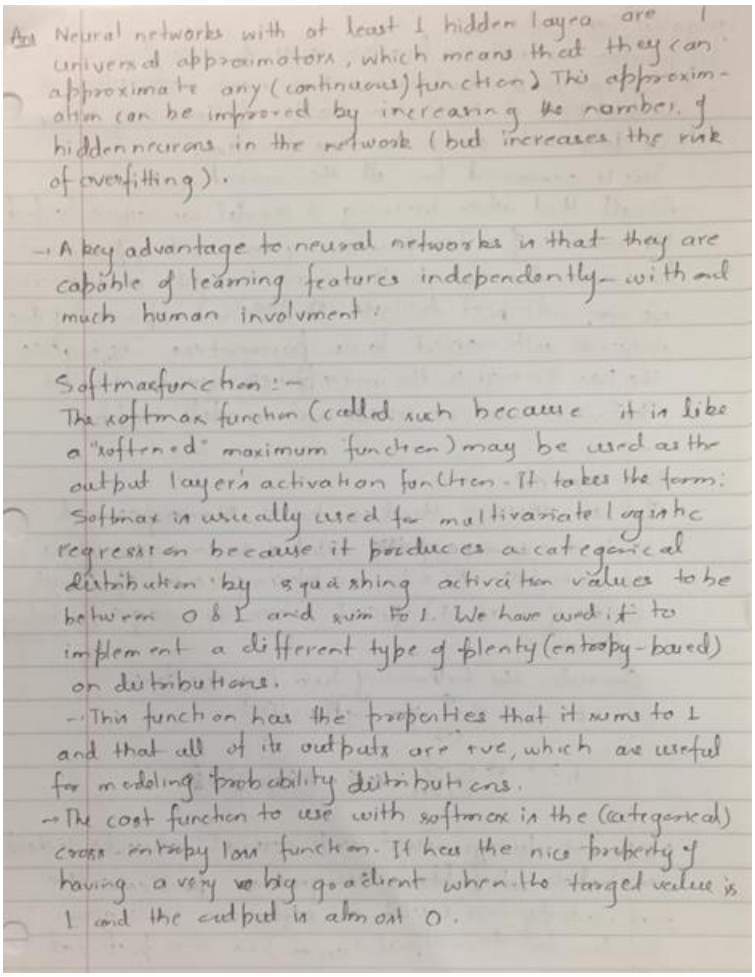
- Let us assume that we have N images and y_i is the label of the image i, where y_i contains $R^{C \times 1}$ - a binary vector of length C (no. of classes) $y_{ic} = 1$ when the image is belonging to class C.

Consider the following two log functions

$$L1 = -\sum_{c=1}^{C} y_{ic} \log P(y_{ic}|D))$$

$$L2 = \sum_{i=1}^{n} \sum_{c=1}^{C} (y_{ic} - P(y_{ic}|D))^2$$

- L2 is not always used with neural networks. Indeed for statistical pattern recognition problems the cross-entropy loss (with a softmax activation function for the output layer) is the preferred option.


Ans Neural networks with at least 1 hidden layer are universal approximators, which means that they can approximate any (continuous) function. This approximation can be improved by increasing the number of hidden neurons in the network (but increases the risk of overfitting).

- A key advantage to neural networks is that they are capable of learning features independently - with and much human involvment.

Softmaxfunction :-
The softmax function (called such because it is like a "softened" maximum function) may be used as the output layer's activation function. It takes the form:
Softmax is usually used for multivariate logistic regression because it produces a categorical distribution by squashing activation values to be between 0 & 1 and sum to 1. We have used it to implement a different type of plenty (entropy-based) on distributions.
- This function has the properties that it sums to 1 and that all of its outputs are +ve, which are useful for modeling probability distributions.
- The cost function to use with softmax is the (categorical) cross-entropy loss function. It has the nice property of having a very big gradient when the target value is 1 and the output is almost 0.

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.

Question 2 (a)

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).

Solution:

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

```
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
- 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.

References

https://github.com/rajivranjanbuff

Question 2 (b)

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

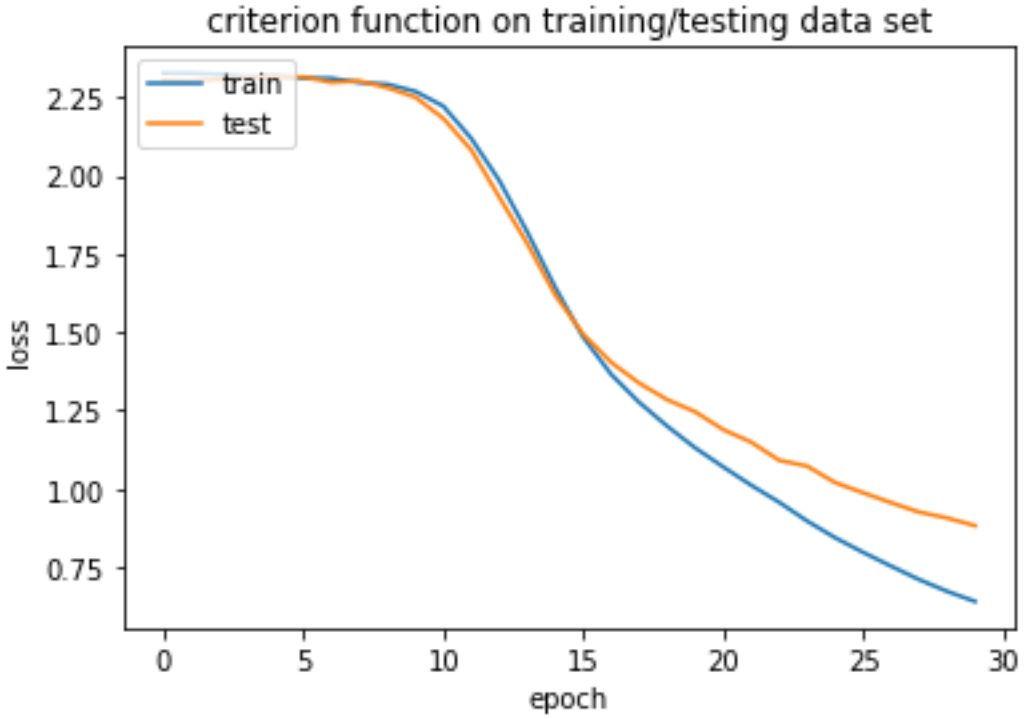
Epoch 28/30
 - 0s - loss: 0.7097 - acc: 0.8140 - val_loss: 0.9253 - val_acc: 0.7130
 - LR: 0.097276

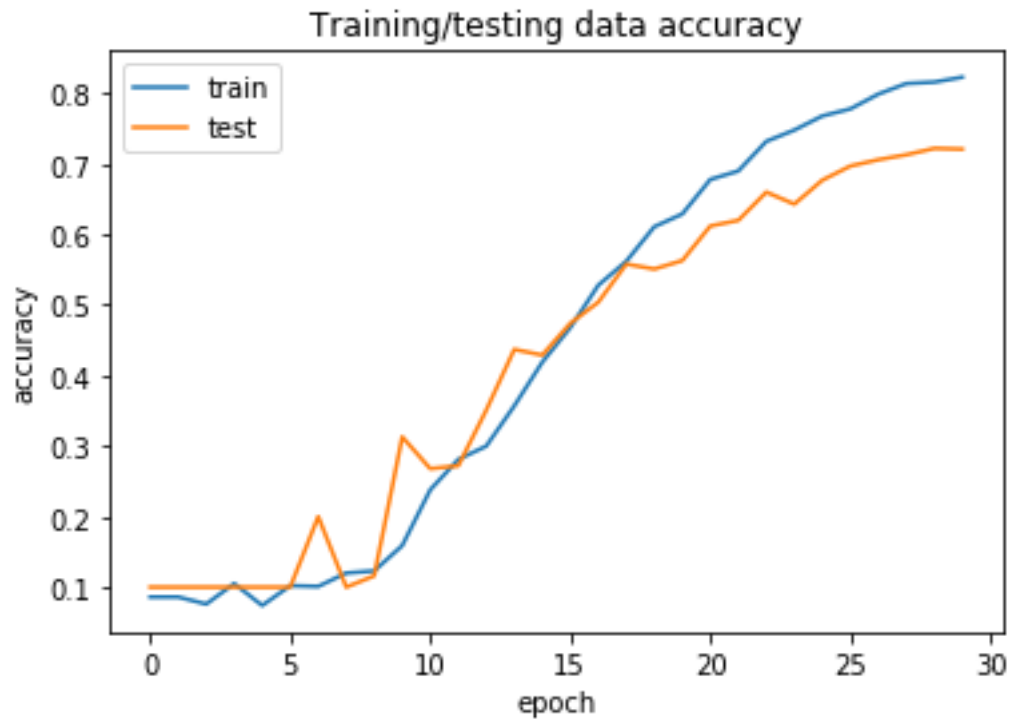
Epoch 29/30

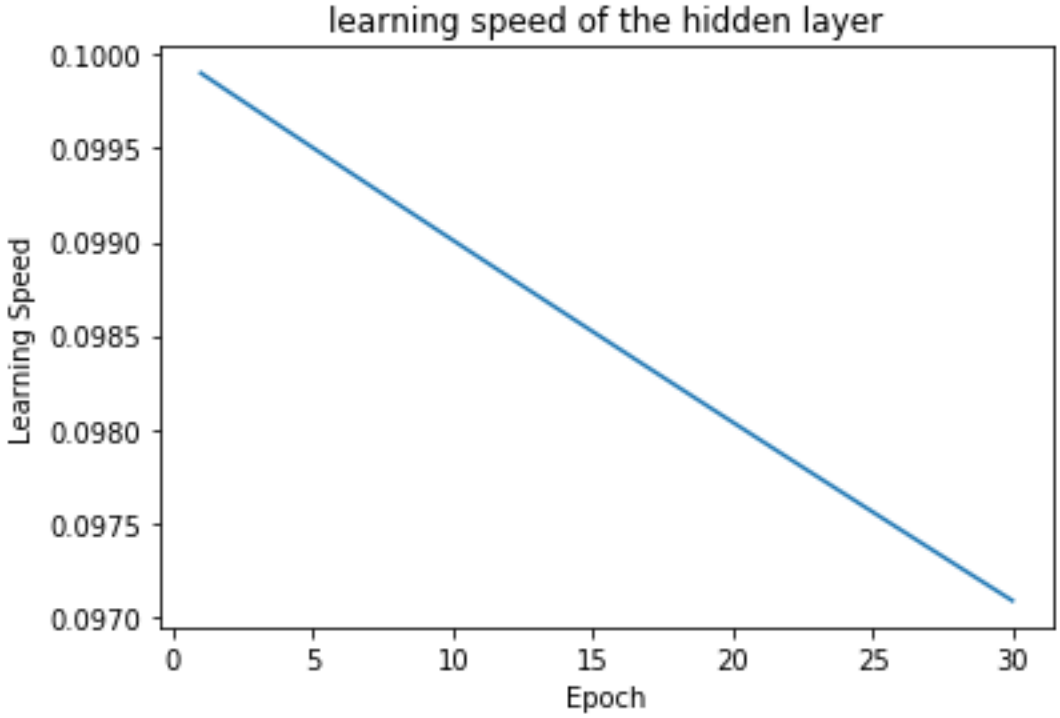
- 0s - loss: 0.6722 - acc: 0.8160 - val_loss: 0.9065 - val_acc: 0.7220
- LR: 0.097182

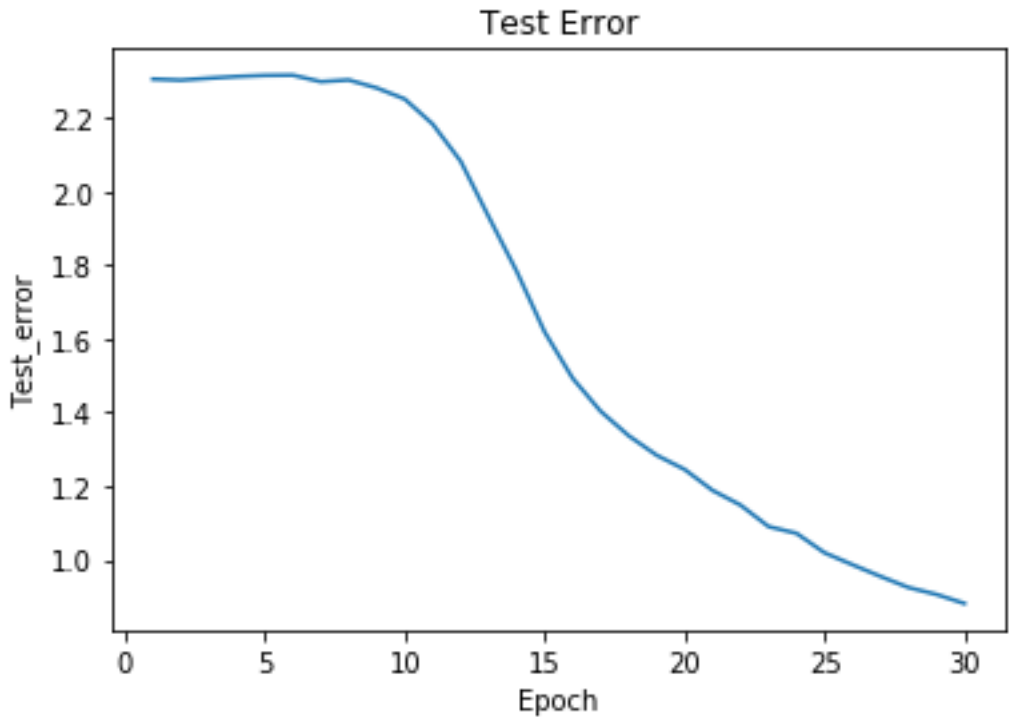
Epoch 30/30
- 0s - loss: 0.6397 - acc: 0.8230 - val_loss: 0.8820 - val_acc: 0.7210
- LR: 0.097087

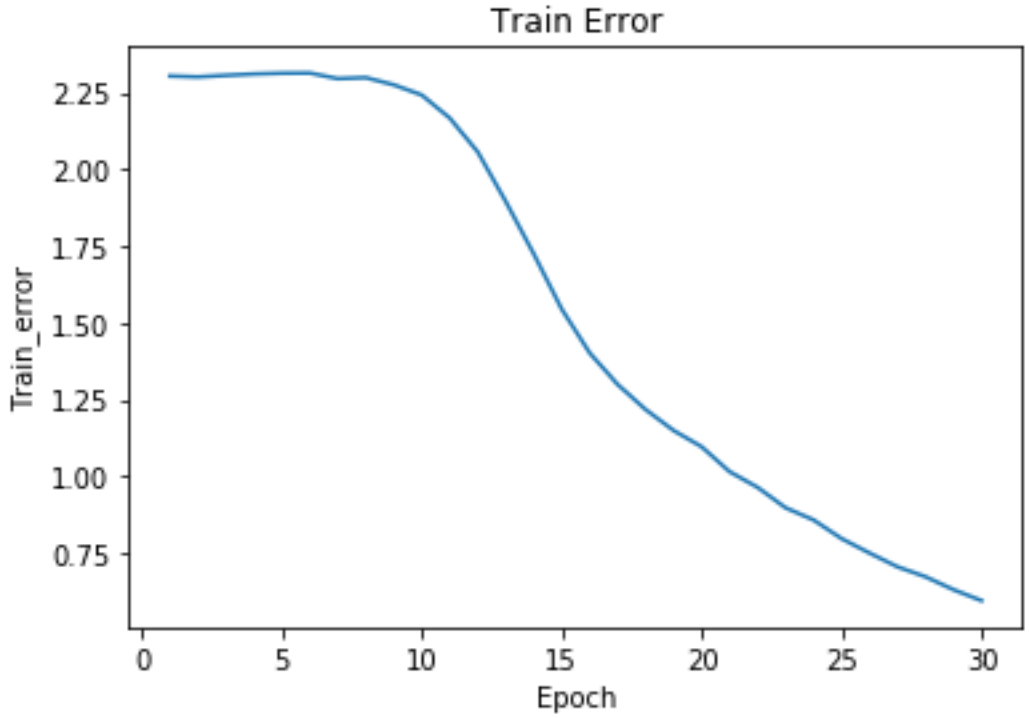
Baseline Error: 27.90%
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])




Training/testing data accuracy
train
test
accuracy
epoch
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
30


learning speed of the hidden layer
Learning Speed
Epoch
0.1000
0.0995
0.0990
0.0985
0.0980
0.0975
0.0970
0
5
10
15
20
25
30





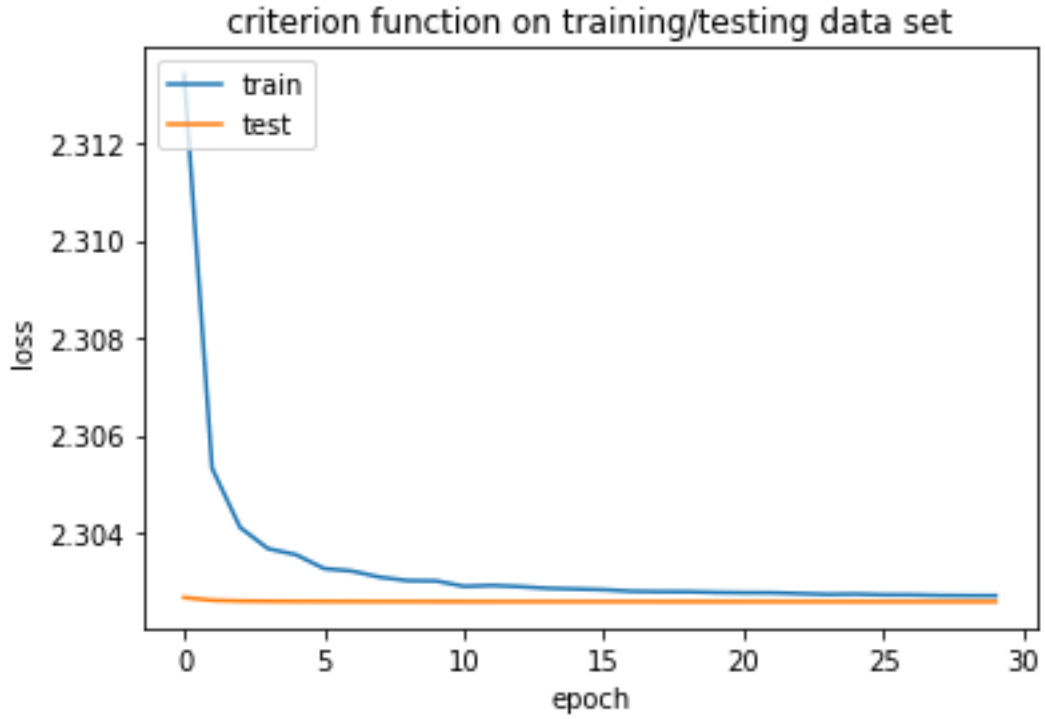
output of three_hidden_layer.py--- three_hidden_layer without the L2 regularization:

Layer (type) Output Shape Param #
==========

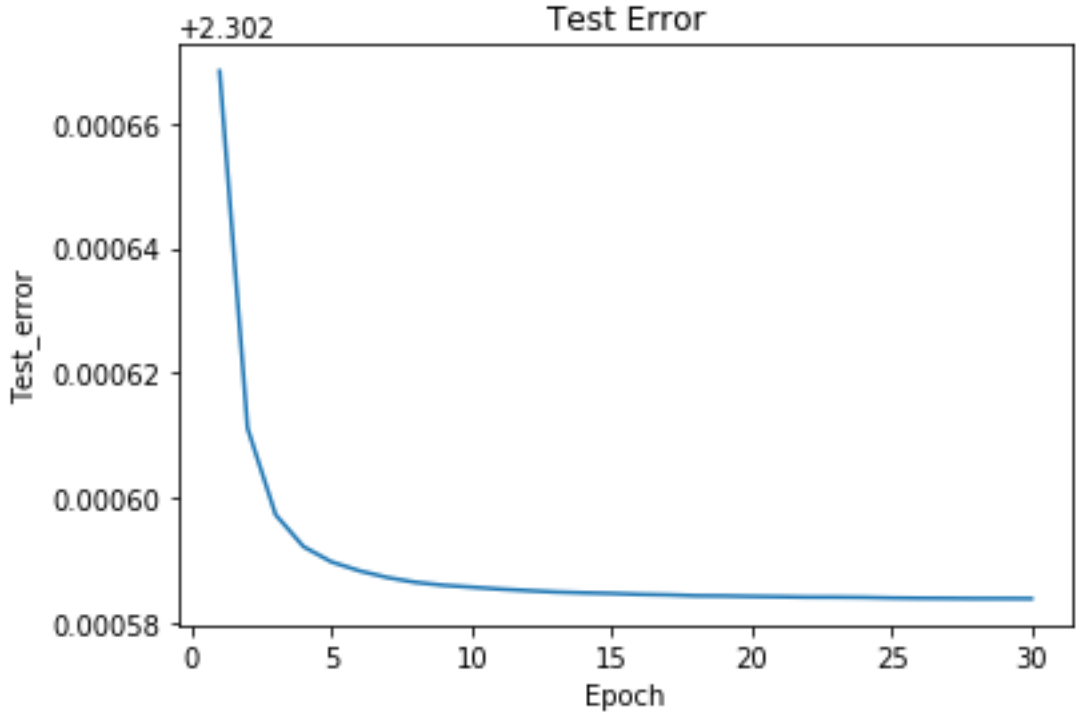
```
dense_36 (Dense)             (None, 30)                23550
_________________________________________________________________
dense_37 (Dense)             (None, 30)                930
_________________________________________________________________
dense_38 (Dense)             (None, 30)                930
_________________________________________________________________
dense_39 (Dense)             (None, 10)                310
=================================================================
Total params: 25,720
Trainable params: 25,720
Non-trainable params: 0
_________________________________________________________________
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
```

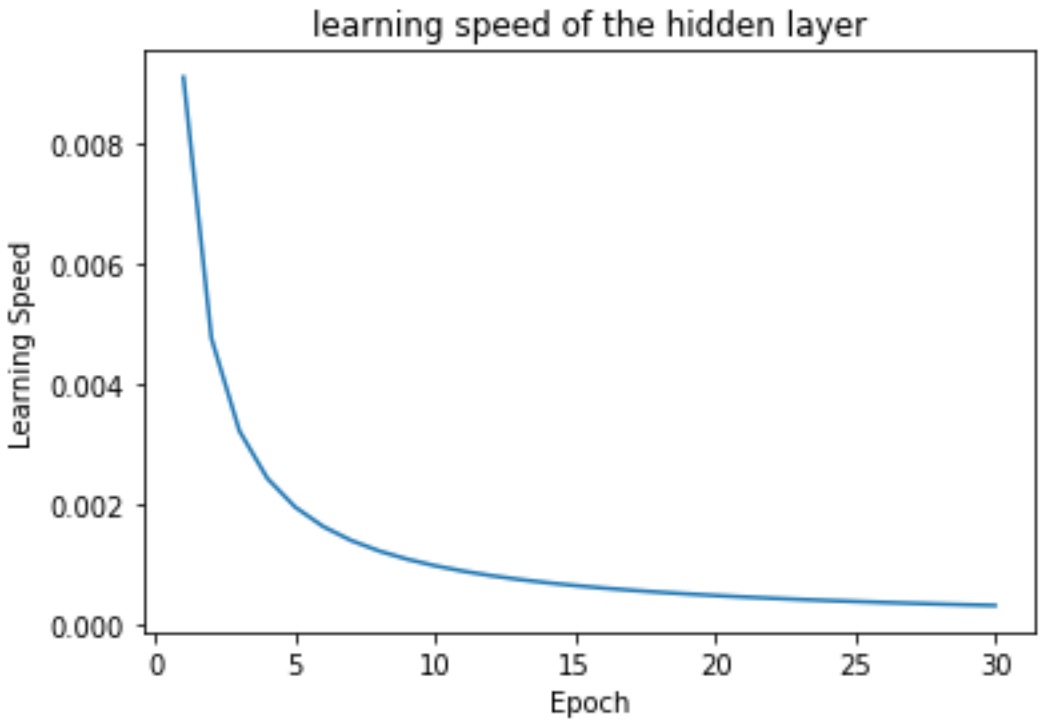
Epoch 12/30
 - 0s - loss: 2.3029 - acc: 0.0870 - val_loss: 2.3026 - val_acc: 0.1000
 - LR: 0.000826
Epoch 13/30
 - 0s - loss: 2.3029 - acc: 0.0780 - val_loss: 2.3026 - val_acc: 0.1000
 - LR: 0.000763
Epoch 14/30
 - 0s - loss: 2.3029 - acc: 0.0830 - val_loss: 2.3026 - val_acc: 0.1000
 - LR: 0.000709
Epoch 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

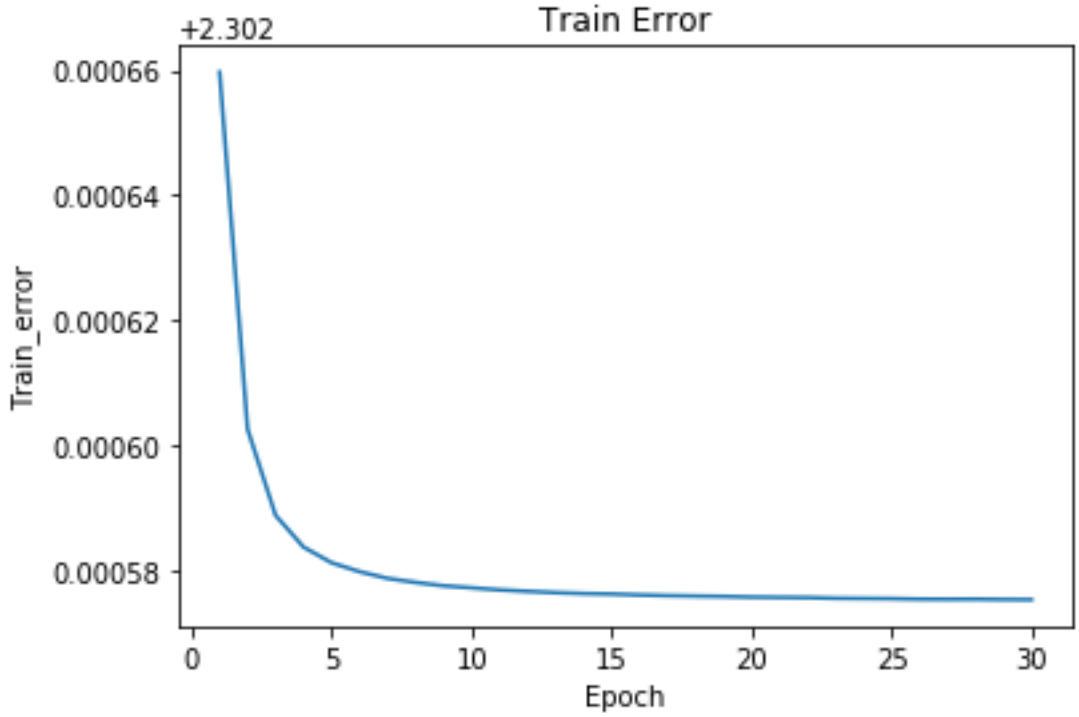
- 0s - loss: 2.3027 - acc: 0.0950 - val_loss: 2.3026 - val_acc: 0.1000
- LR: 0.000369
Epoch 28/30
- 0s - loss: 2.3027 - acc: 0.0890 - val_loss: 2.3026 - val_acc: 0.1000
- LR: 0.000356
Epoch 29/30
- 0s - loss: 2.3027 - acc: 0.0980 - val_loss: 2.3026 - val_acc: 0.1000
- LR: 0.000344
Epoch 30/30
- 0s - 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'])




Training/testing data accuracy
train
test
0.100
0.095
0.090
0.085
0.080
0.075
accuracy
0
5
10
15
20
25
30
epoch


Test Error
+2.302
0.00066
0.00064
0.00062
0.00060
0.00058
Test_error
0
5
10
15
20
25
30
Epoch


learning speed of the hidden layer
Learning Speed
0.008
0.006
0.004
0.002
0.000
0
5
10
15
20
25
30
Epoch


Train Error
+2.302
Train_error
0.00066
0.00064
0.00062
0.00060
0.00058
0
5
10
15
20
25
30
Epoch

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

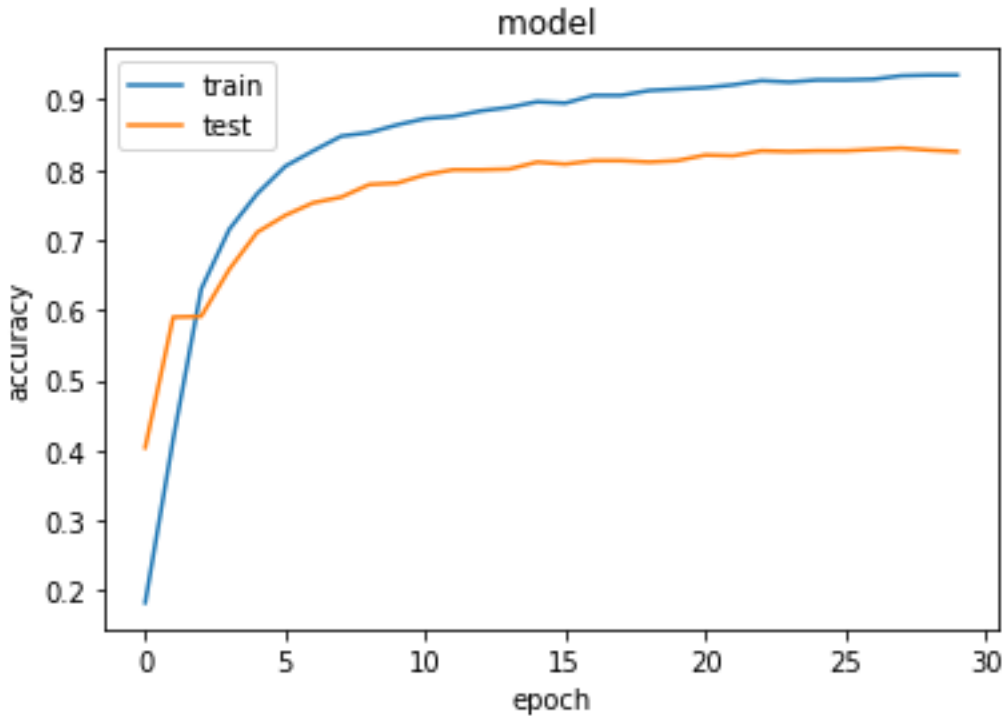
Epoch 27/30
- 0s - loss: 0.3283 - acc: 0.9290 - val_loss: 0.5667 - val_acc: 0.8290
- LR: 0.027027

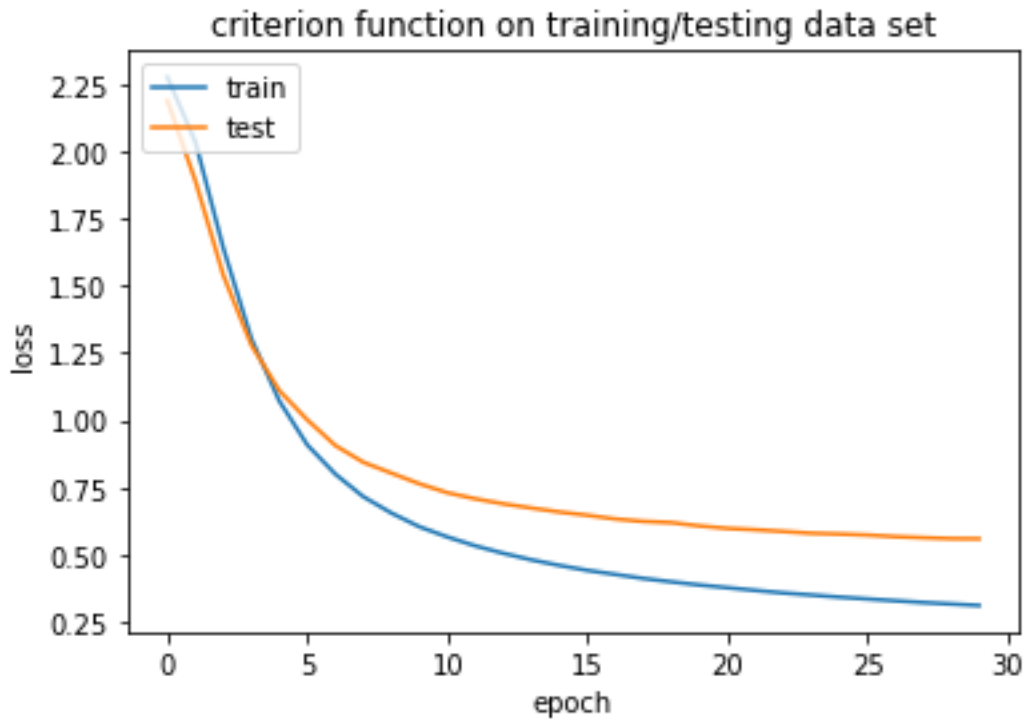
Epoch 28/30
- 0s - loss: 0.3216 - acc: 0.9340 - val_loss: 0.5628 - val_acc: 0.8310
- LR: 0.026316

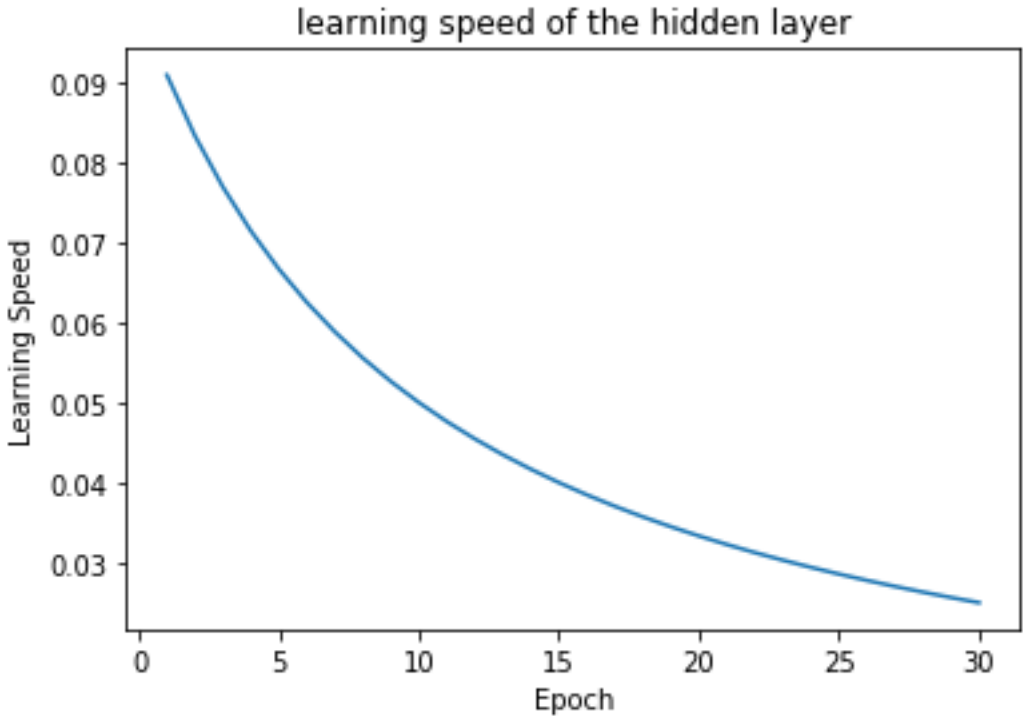
Epoch 29/30
 - 0s - loss: 0.3162 - acc: 0.9350 - val_loss: 0.5594 - val_acc: 0.8280
 - LR: 0.025641

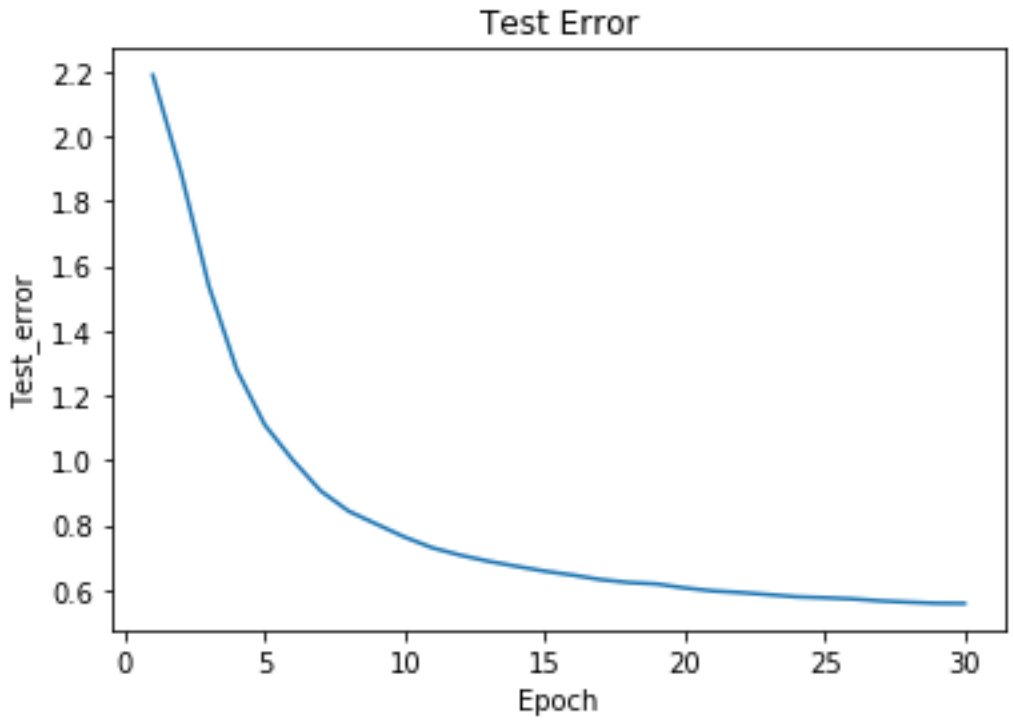
Epoch 30/30
 - 0s - loss: 0.3107 - acc: 0.9350 - val_loss: 0.5588 - val_acc: 0.8260
 - LR: 0.025000

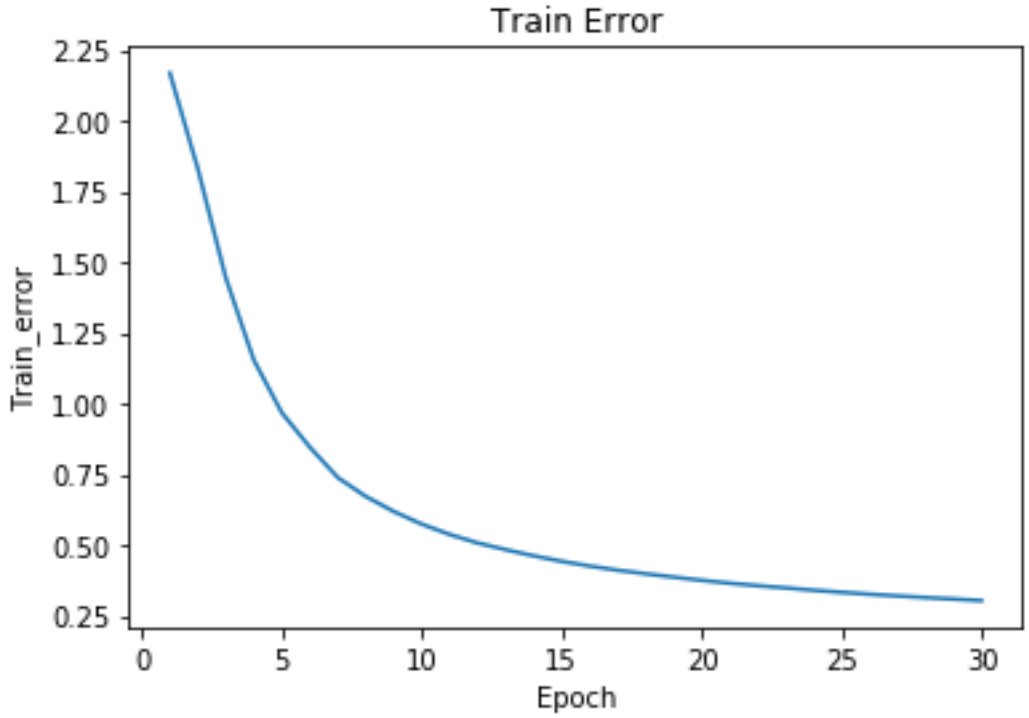
Baseline Error: 17.40%
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])




criterion function on training/testing data set
train
test
loss
epoch
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
5
10
15
20
25
30


learning speed of the hidden layer
Learning Speed
Epoch
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0
5
10
15
20
25
30





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

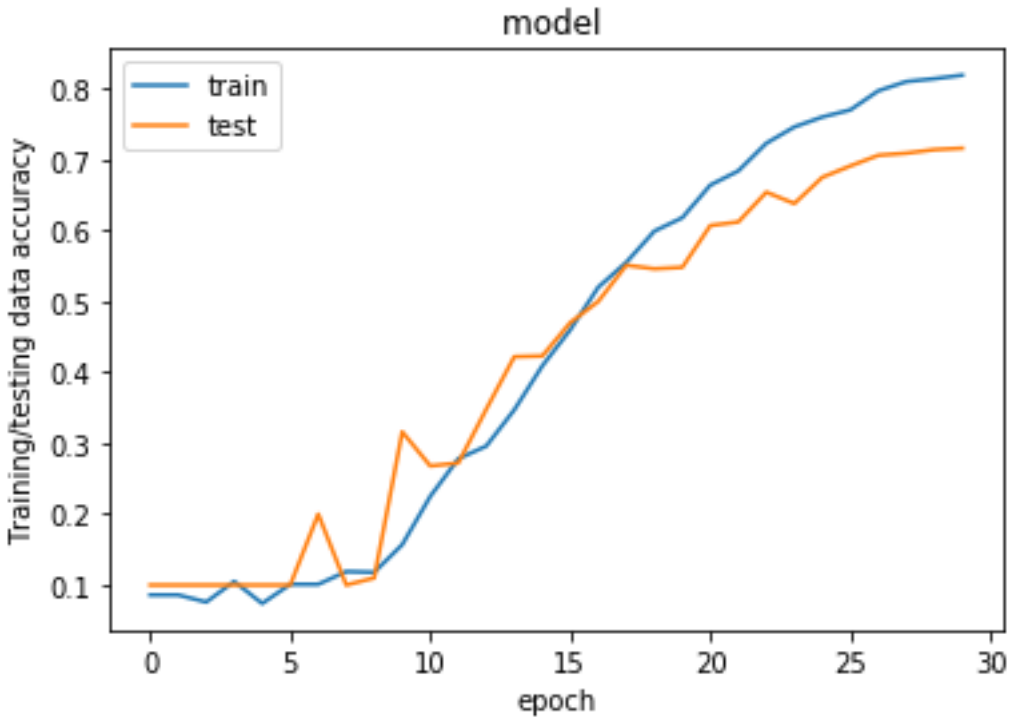
Epoch 27/30
- 0s - loss: 0.7853 - acc: 0.7970 - val_loss: 0.9851 - val_acc: 0.7060
- LR: 0.097371

Epoch 28/30
- 0s - loss: 0.7427 - acc: 0.8100 - val_loss: 0.9552 - val_acc: 0.7090
- LR: 0.097276

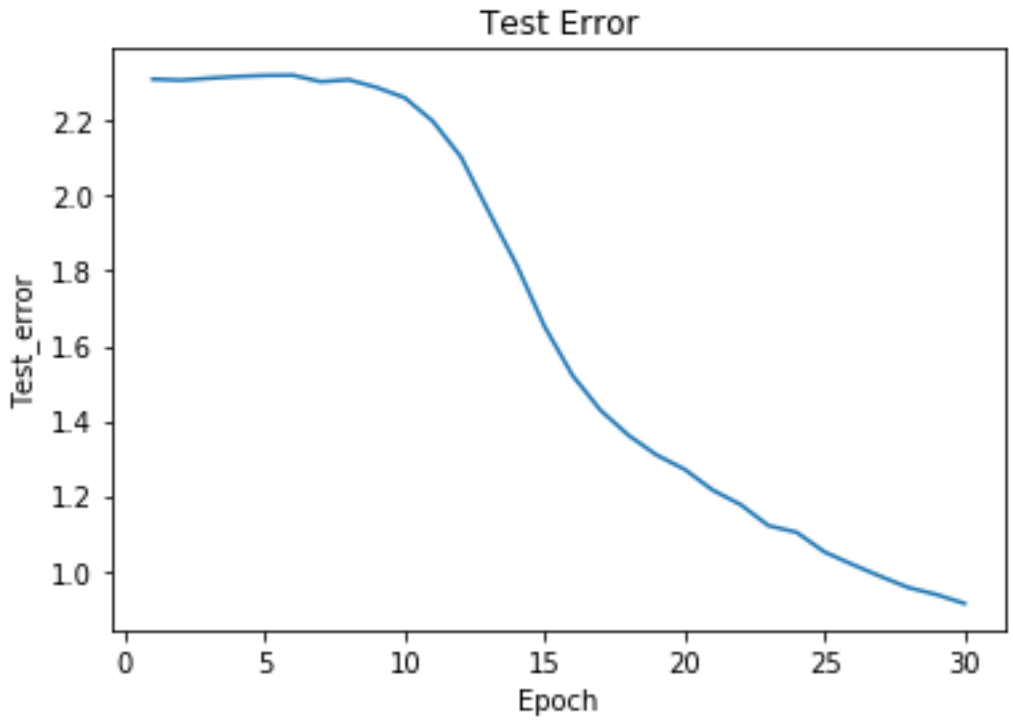
Epoch 29/30
- 0s - loss: 0.7059 - acc: 0.8140 - val_loss: 0.9366 - val_acc: 0.7140
- LR: 0.097182

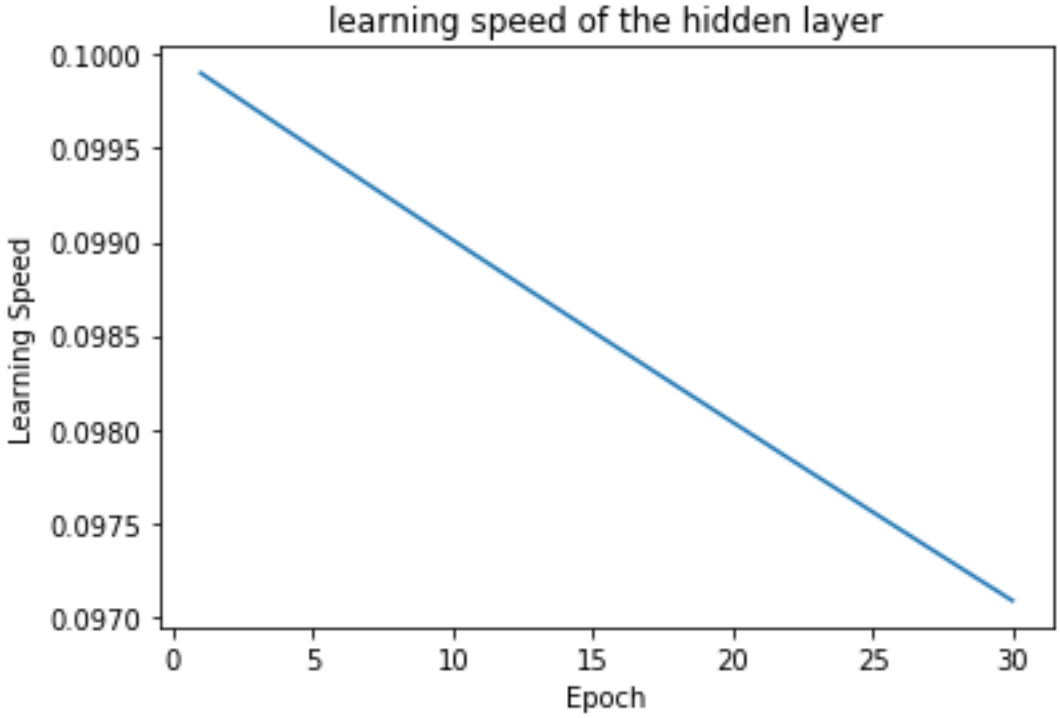
Epoch 30/30
- 0s - loss: 0.6746 - acc: 0.8190 - val_loss: 0.9129 - val_acc: 0.7160
- LR: 0.097087

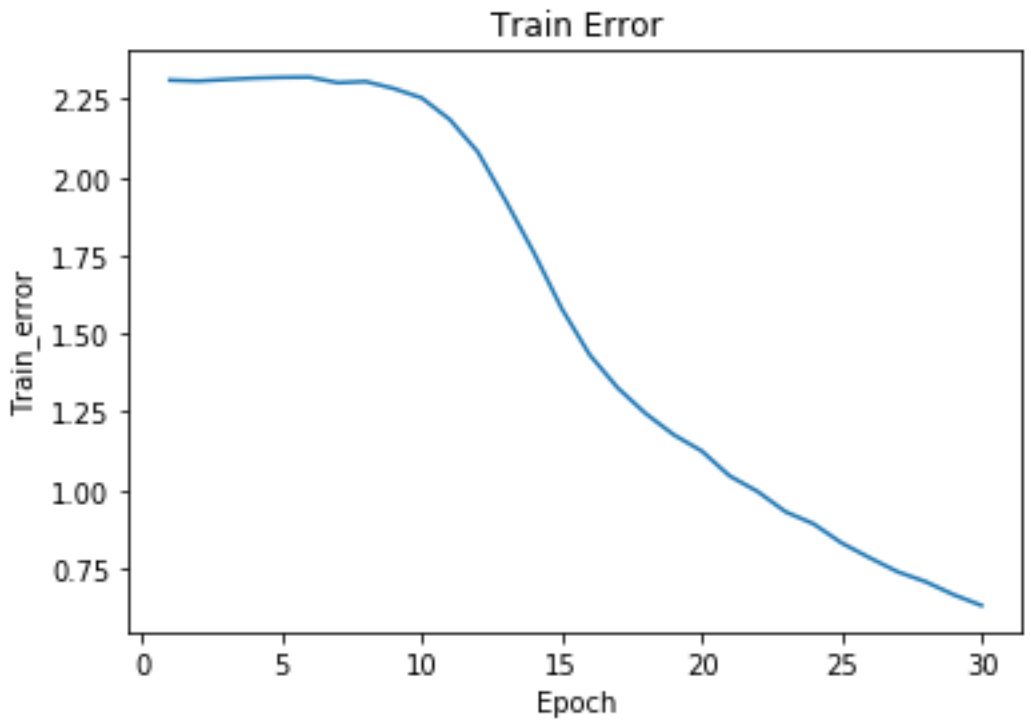
Baseline Error: 28.40%
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])


model
train
test
Training/testing data accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25
30
epoch


criterion function on training/testing data set
train
test
loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0
5
10
15
20
25
30
epoch


Test Error
Test_error
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
5
10
15
20
25
30
Epoch


learning speed of the hidden layer
Learning Speed
0.1000
0.0995
0.0990
0.0985
0.0980
0.0975
0.0970
0
5
10
15
20
25
30
Epoch



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

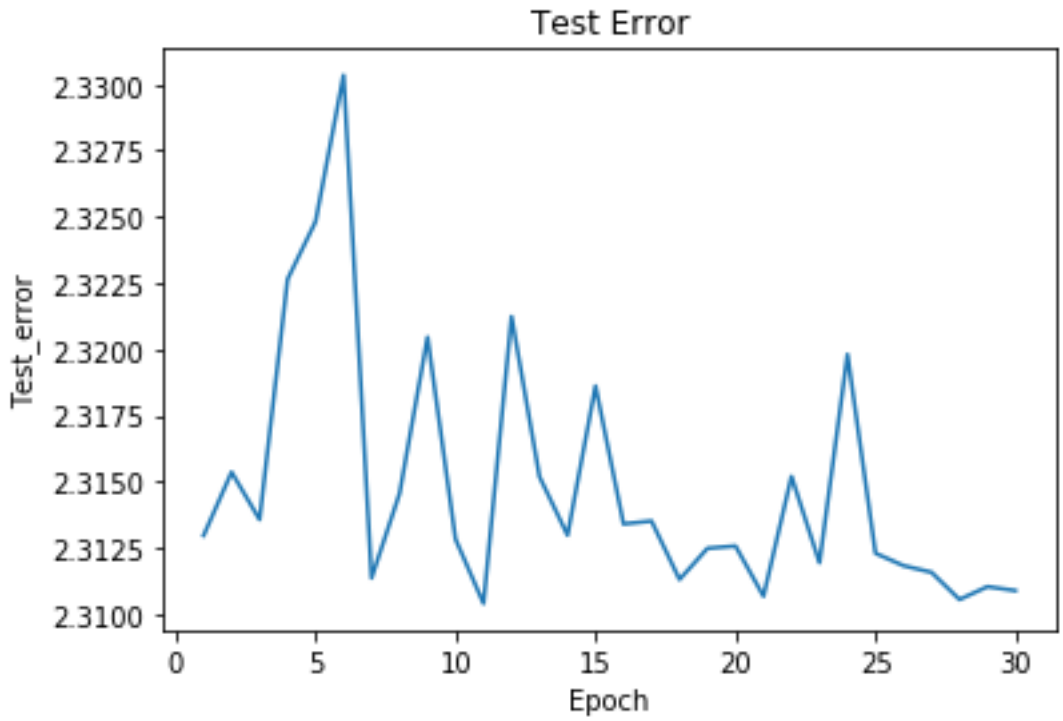
Epoch 29/30
 - 0s - loss: 2.3185 - acc: 0.0900 - val_loss: 2.3110 - val_acc: 0.1000
 - LR: 0.077519

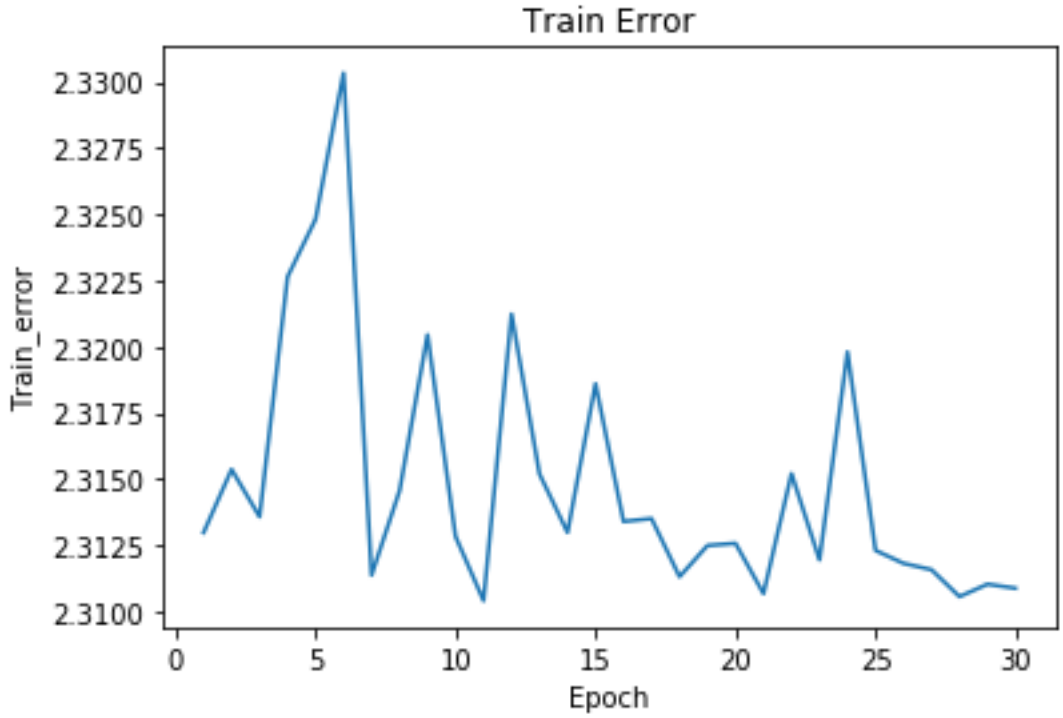
Epoch 30/30
 - 0s - loss: 2.3189 - acc: 0.0860 - val_loss: 2.3109 - val_acc: 0.1000
 - LR: 0.076923

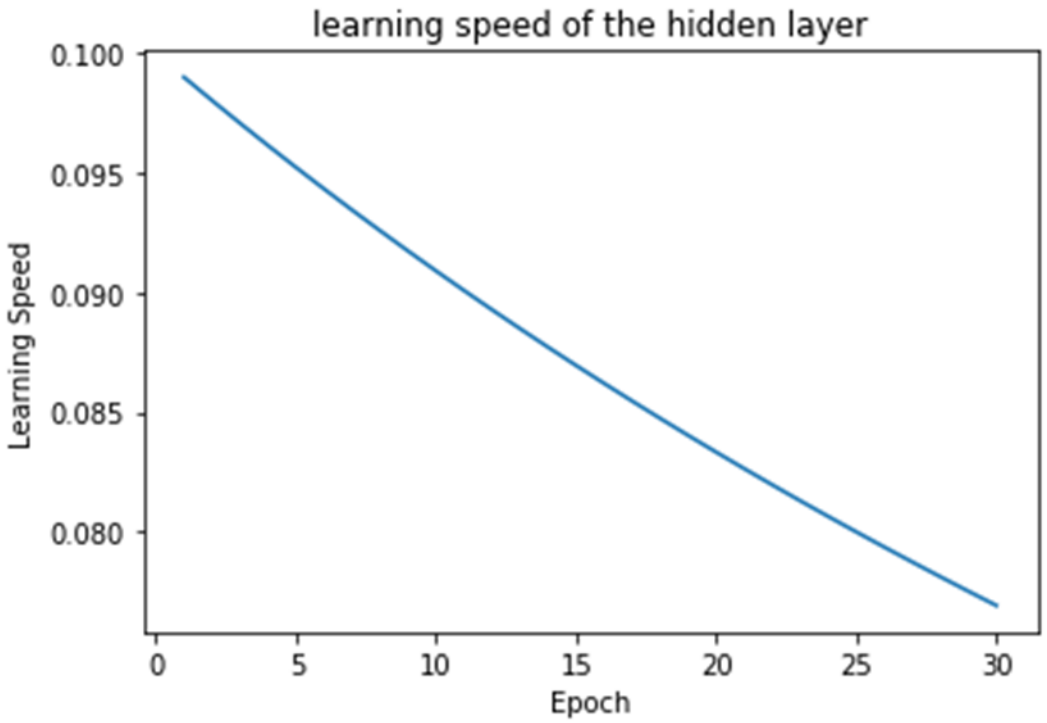
Baseline Error: 10.00%
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])


Training/testing data accuracy
train
test
accuracy
epoch
0.115
0.110
0.105
0.100
0.095
0.090
0.085
0.080
0
5
10
15
20
25
30


criterion function on training/testing data set
train
test
loss
epoch
2.335
2.330
2.325
2.320
2.315
2.310
0
5
10
15
20
25
30


Test Error
Test_error
2.3300
2.3275
2.3250
2.3225
2.3200
2.3175
2.3150
2.3125
2.3100
0
5
10
15
20
25
30
Epoch


Train Error
Train_error
2.3300
2.3275
2.3250
2.3225
2.3200
2.3175
2.3150
2.3125
2.3100
0
5
10
15
20
25
30
Epoch



Question 2 (c)

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**

(c) Regularize the training of neural network through augment yarn 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.

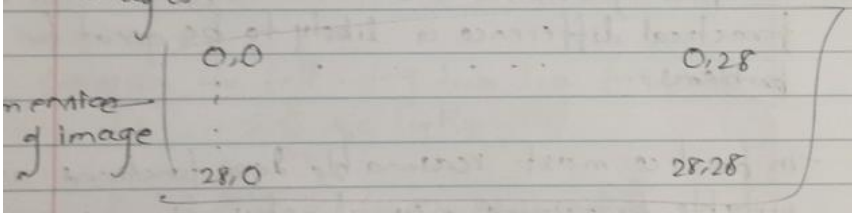
Soln:- In python, we can rotate an image by using the function

out = rotate(input_img, ~~image~~degree_rd, reshape = False)

→ degree_rot = 3 (counter-clockwise rotation of 3 degree

degree_rot = 357 (360-3)
↳ clockwise rotation of 3 degree

shifting in 8 directions: for each of 1000 images:



The shifting is as follows:-

0,0 0,28

28,0 28,28

left shift ← rows, 14, 15, 16

shift in this direction

right shift rows 22, 23, 24 → (direction)

top shift ↑ → columns 17, 18, 19

bottom shift → ↓ columns 14, 15, 16

4 directions,

4 directions

for this firstly both the diagonals were found out & then they were rotated in both the directions. (2+2) = 4 directions

→ Hence the image was counterclockwise rotated by 3 degree & 3 pixels shifted in 8 direction; right, left, top, bottom, top left, top right, bottom left, bottom right → Hence, these were implemented

left ↑ top right ↓ bottom

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

 - 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.