Assignment 2

Neural Networks and Deep Learning

CSCI 5922

Spring 2017

Student:

Bo Cao

boca7588@colorado.edu

Github: https://github.com/BryanBo-Cao/neuralnets-deeplearning

Part 1

(1a) Report the training and test set performance in terms of % examples classified correctly.

Me:
1) learning rate: 0.0001, online learning, 1000 training epochs
Initial input to hidden layer weights: [1.2991697972634182, 0.8961023361530125, 0.03206244212586351, 0.19105435698890227, 0.8210419596211753, 1.2328338604634252]
Trained input to hidden layer weights: [1.3806429847311219, -0.837629335296044, 10.758661847445273, 6.2205720692178472, 0.27167487849155786, -4.302545509560737]

Epoch_when_highest_train_accuracy: 998
Train_accuracy_when_highest_train_accuracy: 0.966105857792
Test_accuracy_when_highest_train_accuracy: 0.940422477441
ws_when_highest_train_accuracy: [1.3860886386405975, -0.83740168462795617, 10.74181016147268, 6.2165876049131423, 0.27311262671962105, -4.3023306543603201]

Epoch_when_highest_test_accuracy: 996
Train_accuracy_when_highest_test_accuracy: 0.965860248066
Test_accuracy_when_highest_test_accuracy: 0.940422477441
ws_when_highest_test_accuracy: [1.3897212975323623, -0.83724813100396811, 10.730556449541272, 6.2139218128952178, 0.27407269494902936, -4.3021849560170331]

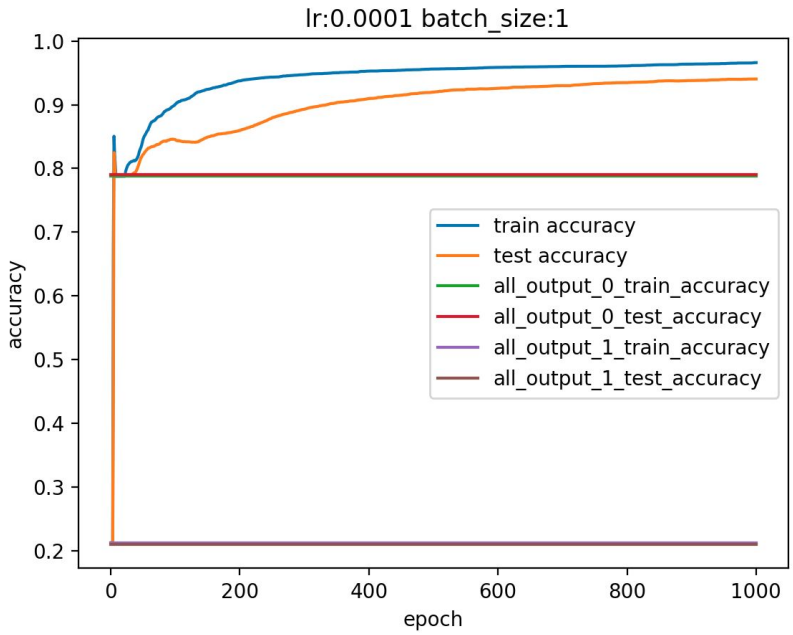

Figure 1.1

2) learning rate: 0.001, online learning, 1000 training epochs
Initial input to hidden layer weights: [1.2624487300153289, 1.7267232989789707, 1.060208249461522, 0.8355883173863075, 1.1747226107524285, 0.9279616314839354]
Trained input to hidden layer weights: [-3.7595122424109153, -0.19467641686682918, 25.772743186166046, 8.7077984078044075, -0.78293983723048022, -4.8693916410650404]

Epoch_when_highest_train_accuracy: 521
Train_accuracy_when_highest_train_accuracy: 0.986737074788
Test_accuracy_when_highest_train_accuracy: 0.989540607055
ws_when_highest_train_accuracy: [-2.3754116811094619, -0.33672394737812461, 21.669812907270252, 7.890459556049163, -0.58737485372929787, -4.5966368033080851]

Epoch_when_highest_test_accuracy: 963
Train_accuracy_when_highest_test_accuracy: 0.986245855336
Test_accuracy_when_highest_test_accuracy: 0.9903609516
ws_when_highest_test_accuracy: [-3.6838971873189039, -0.20324427549112237, 25.547374022565098, 8.6591750128798246, -0.77228859699463992, -4.8518143578050985]

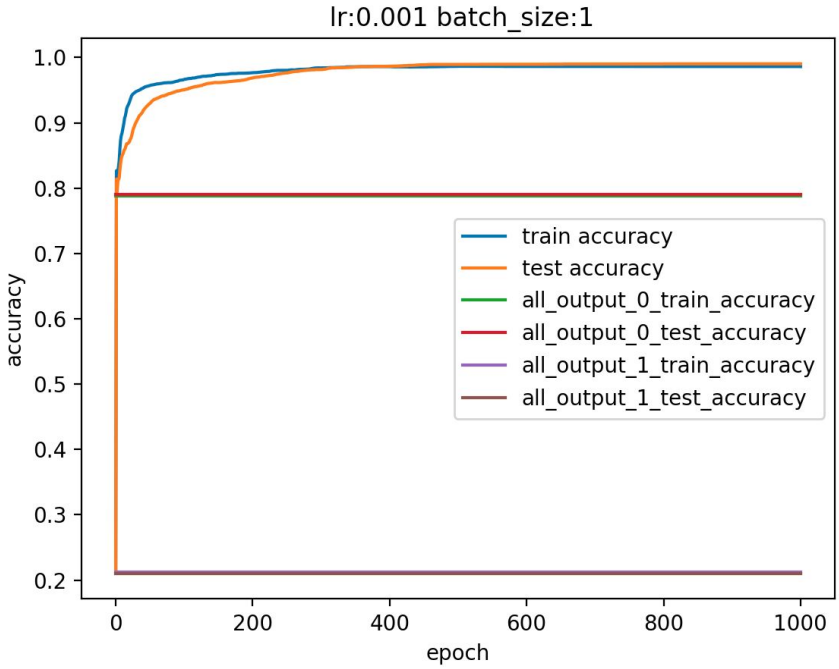

Figure 1.2

Part 2

In my neural network, I used sigmoid and threshold (output 1 when sigmoid(input) >= 0.5; otherwise output 0) as my perceptron function. All weights are initialized randomly to 0 to 1. In addition, I normalized 5 inputs to range from 0 to 1.

(2a) Decide what error function you wish to use. Two obvious candidates are squared error and cross entropy. Report which you have picked.

Me: I used **Mean Squared Error** as loss function.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

(2b) As a way of verifying that your network learns something beyond prior statistics of the training set, let's compute a measure of baseline performance. As a measure of baseline, use the training set to determine the constant output level of the network, call it C, that will minimize your error measure. That is, assume your net doesn't learn to respond to the inputs, but rather gives its best guess of the output without regard to the input. Then for any example where the target is 0 the network will output C and for any example where the target is 1 the network will output C. Using your error measure, solve for C and compute the baseline error. Report the baseline error.

Me:
In terms of error, the training data set, when my network always outputs 0, my error function's result is **0.106164804126**, I refer it to **baseline_error0**; when my network always outputs 1, my error function's result is **0.393835195874**, I refer it to **baseline_error1**. In the graph with respect to error, these error baselines are displayed horizontally.

In addition, accuracies in terms of train set and test set are calculated as baselines. When my network always outputs 0, the accuracies are
Train accuracy with all outputs 0: 0.787670391748
Test accuracy with all outputs 0: 0.789889253486

When my network always outputs 1, the accuracies are
Train accuracy with all outputs 1: 0.212329608252
Test accuracy with all outputs 1: 0.210110746514
In the accuracy graph, these accuracy baselines are shown horizontally.

(2c) Using a network with H=5 hidden units, and mini-batches of size N=100, select a learning rate (or a learning rate schedule) that results in fairly consistent drops in error from one epoch to

the next, make a plot of the training error as a function of epochs. On this graph, show a constant horizontal line for the baseline error. If your network doesn't drop below this baseline, there's something going awry. For now, train your net until you're pretty sure the training error isn't dropping further (i.e., a local optimum has been reached).

Me:

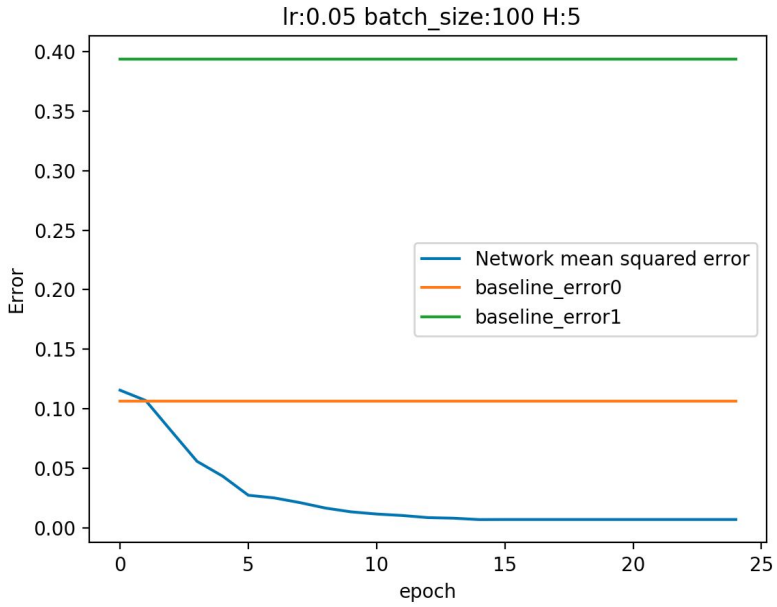

Figure 2c.1

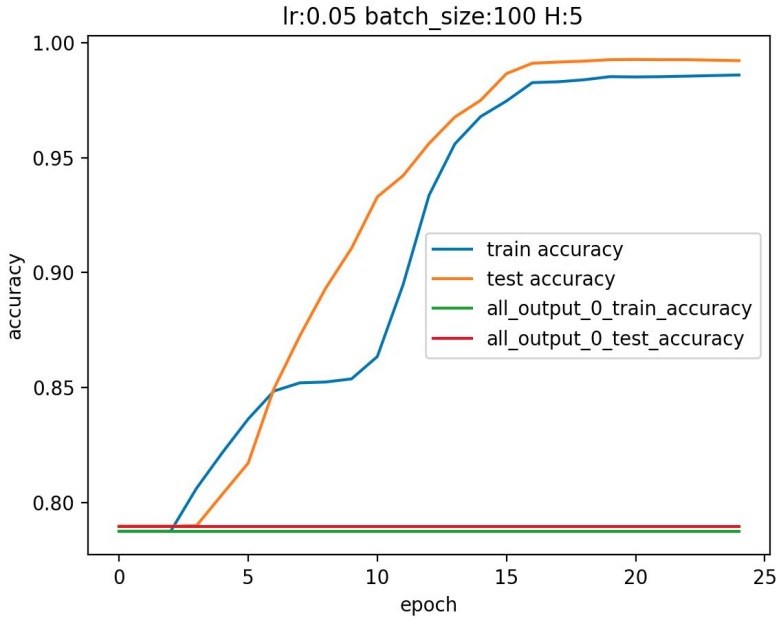

Figure 2c.2

(2d) Report the learning rate (or learning rate schedule) you used to produce the plot in (2c)

Me: learning rate : 0.05

(2e) Report training and test set performance in terms of % examples classified correctly.

Me:
Epoch_when_highest_train_accuracy: 25
Train_accuracy_when_highest_train_accuracy: 0.985877440747
Test_accuracy_when_highest_train_accuracy: 0.992104183757
ws0_when_highest_train_accuracy: [[0.33838217 0.3881548 -4.53720427 -1.10310615 -0.16357794]
 [0.78407736 0.17629907 -5.00088632 -1.1933741   0.01475673]
 [1.32228992 -0.01735016 -4.66660751 -2.01058157 -0.71875963]
 [-0.71671547 -0.14590979 3.18500184 0.14091834 0.44837152]
 [-1.5495697 -0.54299958 5.06966183 1.16505855 0.61470076]]
ws1_when_highest_train_accuracy: [-5.39360016 -5.68007556 -6.02056197 2.62735282 4.73490807]

Epoch_when_highest_test_accuracy: 21
Train_accuracy_when_highest_test_accuracy: 0.985017806705
Test_accuracy_when_highest_test_accuracy: 0.992616899098
ws0_when_highest_test_accuracy: [[0.32909415 0.30184285 -4.37694707 -1.08127132 -0.15539483]
 [0.785861   0.10644466 -4.82281477 -1.16608186 0.0460361]
 [1.29410815 -0.05240045 -4.52316356 -1.91679273 -0.64935712]
 [-0.73884414 -0.14859609 3.07398891 0.09124413 0.40325455]
 [-1.55731143 -0.52151251 4.89510269 1.10610394 0.55355751]]
ws1_when_highest_test_accuracy: [-5.18889757 -5.47077769 -5.80249932 2.56436996 4.61297094]

I used the test set performance as the main way to evaluate the network, when test accuracy reached the highest accuracy during training, both train accuracy and test accuracy were recorded, they are shown above in the **21st** epoch: **Train Accuracy: 98.50%**, **Test Accuracy: 99.26%**

(2f) Now train nets with varying size, H, in {1, 2, 5, 10 20}. You may have to adjust your learning rates based on H, or use one of the heuristics in the text for setting learning rates to be independent of H. Decide when to stop training the net based on training set performance.

Make a plot, as a function of H, of the training and test set performance in terms of % examples classified correctly.

Me:
The plot is shown below. X-axis is the number of hidden layer units, y-axis is the accuracy. Train accuracy is shown in blue while test accuracy is displayed in red.

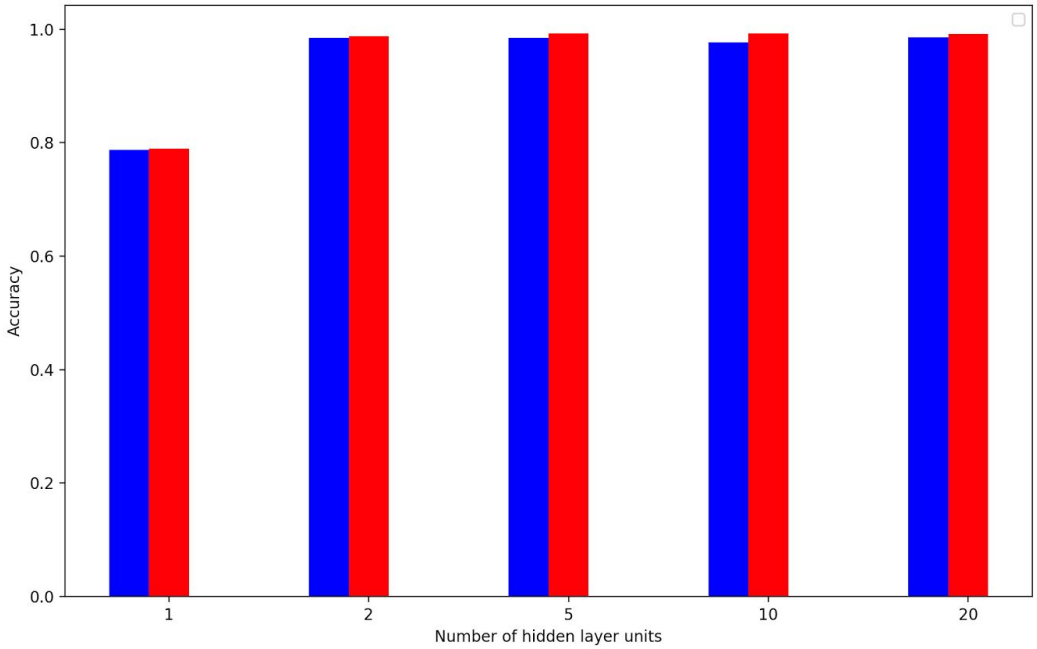

Figure 2f.1

In assignment 1 I used early stopping, when the validation error starts to increase even if the train error is still decreasing, to decide to terminate the training of my network. In this assignment I simply stopped training after a number of epochs, even when test accuracy decreases. The reason for doing this is to explore the performance more.

H:2
Epoch_when_highest_train_accuracy: 71
Train_accuracy_when_highest_train_accuracy: 0.986245855336
Test_accuracy_when_highest_train_accuracy: 0.950164068909
ws0_when_highest_train_accuracy: [[4.06135374 2.03665981 -16.39022483 -5.95969643 -0.1678452]
[0.60399639 0.53732088 -2.14148058 -1.55148233 0.52075169]]
ws1_when_highest_train_accuracy: [-9.2011993 3.20949172]

Epoch_when_highest_test_accuracy: 70

Train_accuracy_when_highest_test_accuracy: 0.984772196979
Test_accuracy_when_highest_test_accuracy: 0.988310090238
ws0_when_highest_test_accuracy: [[3.19525603 1.47074095 -16.49167304 -6.27496872 -0.64755319]
[-0.5006867 -0.36333767 -2.82919556 -2.41187413 -0.34843673]]
ws1_when_highest_test_accuracy: [-8.92387799 2.81826245]

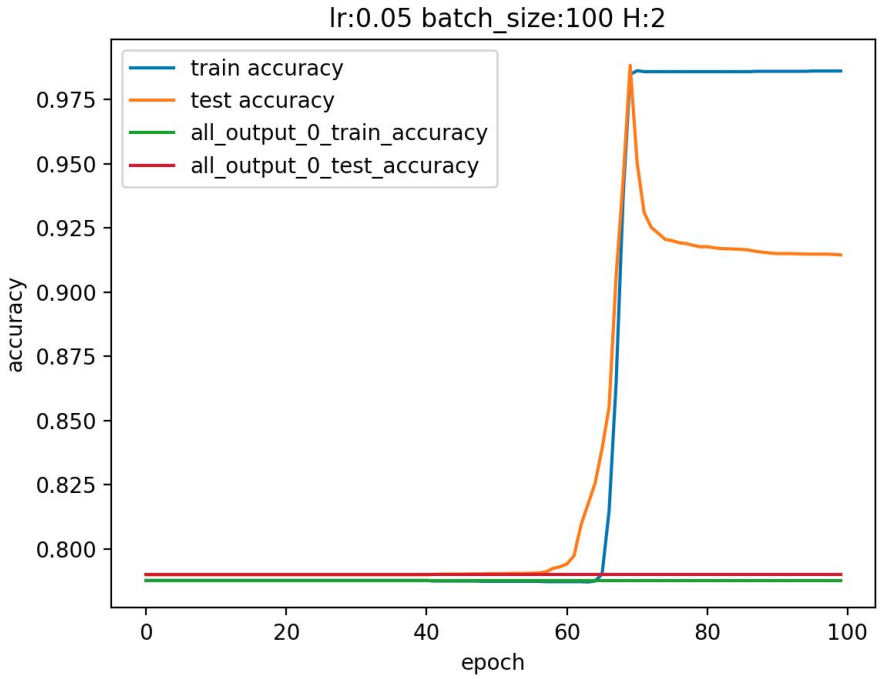

Figure 2f.2

H:5
Results of H:5 please see (2c) to (2e)

H:10
Epoch_when_highest_train_accuracy: 25
Train_accuracy_when_highest_train_accuracy: 0.983912562937
Test_accuracy_when_highest_train_accuracy: 0.992104183757
ws0_when_highest_train_accuracy: [[1.48446569 0.58249199 -6.18665955 -1.36505989 -0.12744187]
[-1.00302462 -0.46759262 -0.11321495 0.12827268 -0.2509269]
[-1.44337603 -0.30367042 4.89580816 1.32020592 0.14051396]
[-0.0134713 0.16374822 -2.96885247 -0.85385384 -0.85451573]
[-0.16249853 -0.57954112 -1.79224282 -0.21483308 -0.89502832]
[-0.80216006 -0.39430893 3.84034636 1.12550802 0.20783004]
[-0.54315323 -0.68865454 -1.13898482 -0.45002514 -0.81655186]
[-0.40359122 -0.36380834 1.26707562 0.05589922 0.06611458]

[-0.40696945 -0.75064945 -0.46240544 -0.02176771 -0.63915999]
[0.62488994 -0.26777986 -3.50963398 -0.58326755 -0.88015159]]
ws1_when_highest_train_accuracy: [-6.90464668 -0.47819616 4.53998505 -3.62977803
-2.25256622 3.51545656
 -1.79932145 0.81463761 -0.89759373 -4.19988302]

Epoch_when_highest_test_accuracy: 19
Train_accuracy_when_highest_test_accuracy: 0.976789880879
Test_accuracy_when_highest_test_accuracy: 0.992821985234
ws0_when_highest_test_accuracy: [[1.49475324e+00   4.10226070e-01 -5.80250749e+00
-1.20740193e+00
   -9.04377971e-02]
 [-9.72734002e-01 -4.63270377e-01 -7.87304021e-02   1.46443208e-01
   -2.38491428e-01]
 [-1.43479872e+00 -2.18892742e-01   4.60927524e+00   1.16739014e+00
    9.80782876e-02]
 [-4.28192737e-02   5.37158898e-02 -2.79756924e+00 -7.39231934e-01
   -8.75541124e-01]
 [-1.58235295e-01 -6.05009213e-01 -1.67620742e+00 -1.20361342e-01
   -8.77595869e-01]
 [-8.10141447e-01 -3.34408200e-01   3.60837423e+00   9.98768581e-01
    1.71398049e-01]
 [-5.40165818e-01 -7.03317079e-01 -1.05309819e+00 -3.67781422e-01
   -7.99481755e-01]
 [-4.05317266e-01 -3.65789960e-01   1.20089478e+00   5.63423946e-03
    4.52883509e-02]
 [-3.71079717e-01 -7.41607233e-01 -4.02848416e-01   2.17920573e-02
   -6.14606720e-01]
 [5.94231563e-01 -3.64700731e-01 -3.30109875e+00 -4.46526538e-01
   -8.77855330e-01]]
ws1_when_highest_test_accuracy: [-6.50893842 -0.40341795 4.25025288 -3.37190022
-2.08093646 3.27232805
 -1.64610476 0.77130859 -0.80213179 -3.94118056]

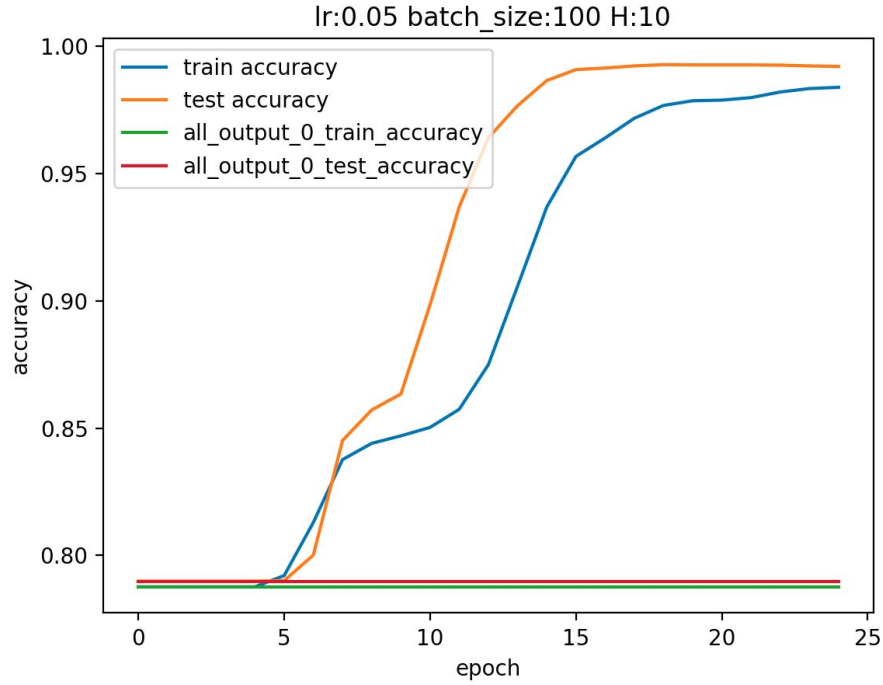

Figure 2f.3

H:20

Epoch_when_highest_train_accuracy: 50

Train_accuracy_when_highest_train_accuracy: 0.98600024561

Test_accuracy_when_highest_train_accuracy: 0.991899097621

ws0_when_highest_train_accuracy: [[-0.75018947 -0.23971662 1.76870874 0.36882569 0.29977896]
[-0.36620192 -0.24564945 0.36310501 0.27124824 0.02057693]
[-0.74552461 -0.23363179 1.27542791 0.04311968 -0.13408172]
[-0.60909848 -0.29975995 2.01634941 0.98415062 -0.0358271]
[-1.01534214 -0.17336032 3.71550308 1.0593093 0.08876372]
[-0.65369272 -0.56654127 3.23348958 0.67950993 0.46280619]
[-0.91652366 0.25263042 2.26795283 0.26357179 0.26163884]
[0.08749728 0.64679947 -3.90186235 -0.85129724 -0.49051382]
[0.55552506 -0.84576378 -2.3435577 -1.59615017 -1.28090619]
[0.25891417 -0.36167858 -3.20151371 -0.49484792 -0.1307034]
[-0.53975722 -0.22931954 0.07265146 0.18065527 -0.30367414]
[0.18827486 0.33096258 -4.18744697 -0.79463968 -0.08949276]
[-0.95988646 -0.09236131 0.62713203 0.44757835 -0.58206665]
[-0.75831653 -0.0727592 0.4079384 -0.22513168 -0.56196395]
[0.76299177 0.1603483 -4.57762232 -0.39231224 -0.68495551]
[-0.7995208 -0.53767914 2.68828968 0.90780437 0.44014328]
[-0.59125766 -0.44884231 -0.72633009 0.045923 -0.06521245]

[0.05888584 -0.60977446 -2.63975271 -0.54416392 -0.52270517]
[-0.68146188 -0.31831717 3.37725112 1.03568752 0.08994817]
[-0.45527906 -0.50220451 -2.22167394 -0.25660375 0.06313487]]
ws1_when_highest_train_accuracy: [1.11078831 -0.13855116 0.51130143 1.38431576
2.88773459 2.44084218
  1.55455914 -4.61902296 -3.70493211 -3.52681542 -0.47786302 -4.73413647
 -0.02480608 -0.33922501 -5.4087294   1.9637939 -1.17687712 -3.47800975
  2.63299168 -2.72018792]

Epoch_when_highest_test_accuracy: 31
Train_accuracy_when_highest_test_accuracy: 0.975193417659
Test_accuracy_when_highest_test_accuracy: 0.992719442166
ws0_when_highest_test_accuracy: [[-7.78837205e-01 -2.26041049e-01   1.63548310e+00
3.04263692e-01
    2.56704393e-01]
 [-2.96172730e-01 -2.25856271e-01   3.91397058e-01   2.75516182e-01
    4.58971553e-02]
 [-7.33500951e-01 -2.26194413e-01   1.21844035e+00   3.49899109e-03
   -1.51850342e-01]
 [-6.47530959e-01 -2.83798712e-01   1.85422862e+00   9.09113207e-01
   -9.11672340e-02]
 [-1.10759834e+00 -1.21595982e-01   3.40232977e+00   9.29753501e-01
   -1.81573404e-02]
 [-7.33939264e-01 -5.22376476e-01   2.96430477e+00   5.65497599e-01
    3.75474014e-01]
 [-9.72701042e-01   2.69659654e-01   2.08167677e+00   1.80287464e-01
    1.99290886e-01]
 [1.43902987e-01   3.62624807e-01 -3.51638792e+00 -7.47738577e-01
   -5.27888272e-01]
 [4.67032539e-01 -7.53504888e-01 -2.14089303e+00 -1.27780701e+00
   -1.07180843e+00]
 [3.61284177e-01 -4.61925478e-01 -2.87114908e+00 -3.70903790e-01
   -5.82821561e-02]
 [-4.63248693e-01 -2.14748884e-01   1.36539282e-01   2.03646225e-01
   -2.69262621e-01]
 [2.93515068e-01   6.26762608e-02 -3.77406767e+00 -6.98551046e-01
   -9.68723608e-02]
 [-9.19242464e-01 -7.95874763e-02   6.33846658e-01   4.43908176e-01
   -5.70489648e-01]
 [-7.01590940e-01 -5.93350876e-02   4.48677242e-01 -2.07453295e-01
   -5.36747588e-01]
 [8.16316372e-01 -7.78454557e-02 -4.11216571e+00 -2.69251262e-01
   -6.42160579e-01]

[-8.71811957e-01 -5.15285583e-01   2.46044092e+00   8.06354762e-01
   3.59516756e-01]
[-5.01168209e-01 -4.53952820e-01 -5.92775388e-01   1.02242636e-01
  -2.04068111e-02]
[1.94756430e-02 -6.93508912e-01 -2.39150639e+00 -3.73215756e-01
  -4.68748256e-01]
[-7.60845947e-01 -2.64369295e-01   3.09439990e+00   9.18705581e-01
  -1.31349538e-03]
[-4.13369874e-01 -6.19179592e-01 -1.98683731e+00 -1.61706656e-01
   6.52517679e-02]]
ws1_when_highest_test_accuracy: [1.04149299 -0.088757   0.52764078 1.26659411
2.63557401 2.20081155
  1.4578524 -4.13435764 -3.25816427 -3.14682596 -0.36473053 -4.25218195
  0.059479   -0.20566996 -4.93585271 1.77669376 -0.98972975 -3.12636831
  2.38722833 -2.40624972]

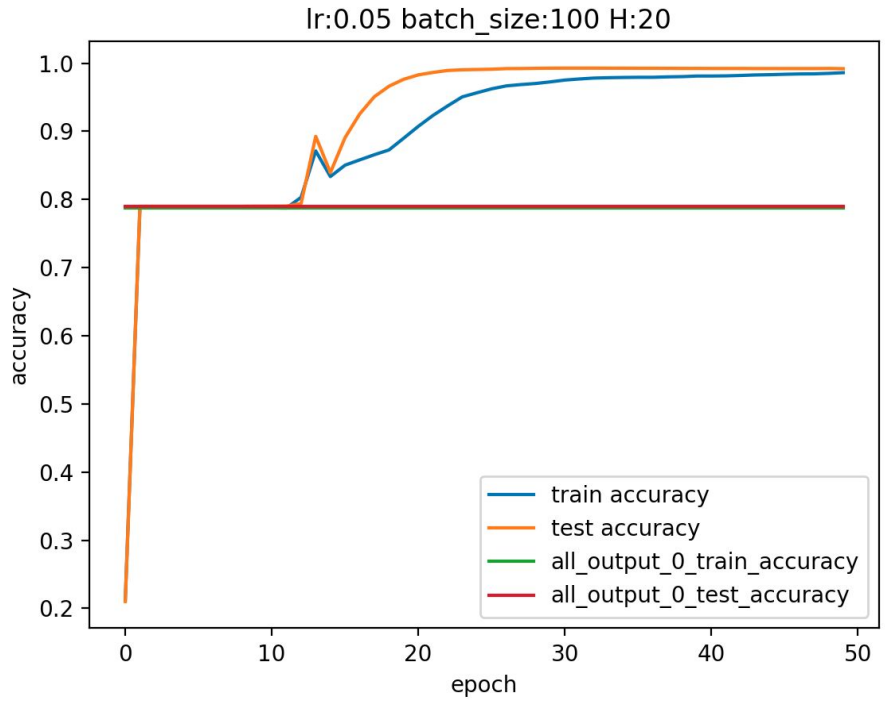

Figure 2f.4

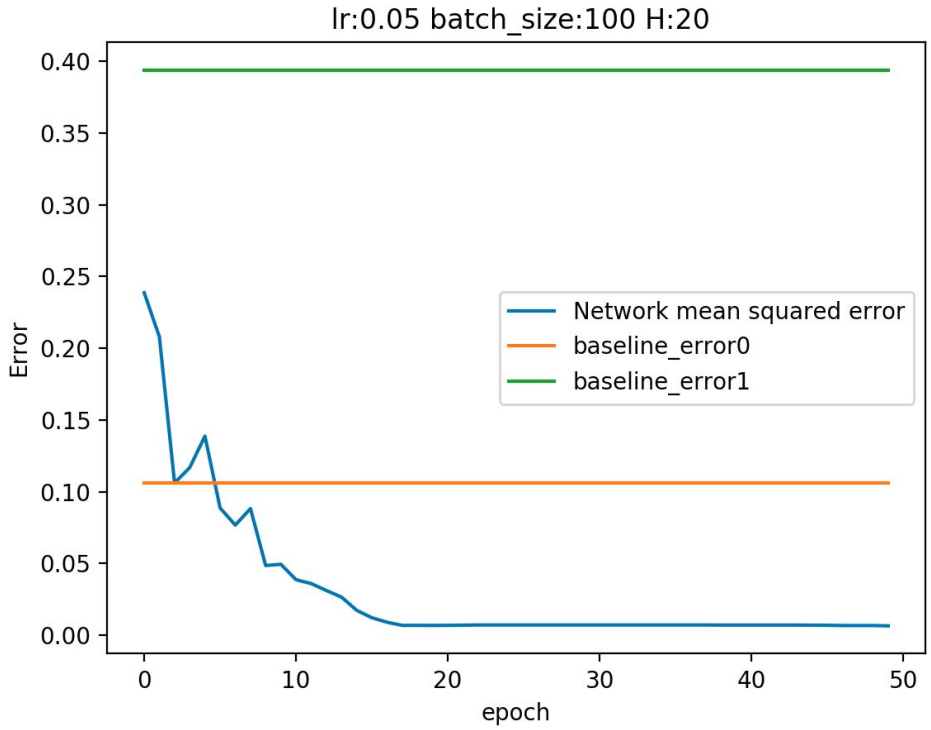

Figure 2f.5

Part 3 (Extra Credit)

(3a) Determine an appropriate representation for the time of day. Describe the representation you used. For example, you might add one unit with a value ranging from 0 to 1 for times ranging from 00:00 to 23:59. Report the representation you selected.

Me:
I map the times ranging from 00:00 to 23:59 to the range from 0 to 1. The figure below showed some of the converted time values, the result can be run by dateParser.py:

```
2015-02-10 09:14:00 time_value:  0.384722222222
2015-02-10 09:15:00 time_value:  0.385416666667
2015-02-10 09:16:00 time_value:  0.386111111111
2015-02-10 09:16:59 time_value:  0.386793981481
2015-02-10 09:17:59 time_value:  0.387488425926
2015-02-10 09:19:00 time_value:  0.388194444444
2015-02-10 09:20:00 time_value:  0.388888888889
```

Figure 3a.1

(3b) Train your net with H=5 hidden and compare training and test set performance to the net you built in (2e)

Me:
Epoch_when_highest_train_accuracy: 85
Train_accuracy_when_highest_train_accuracy: 0.988333538008
Test_accuracy_when_highest_train_accuracy: 0.951907301066
ws0_when_highest_train_accuracy: [[0.26187131 -0.65453698 -1.86671445 -1.96042975
-1.02701164
  -0.25220492]
 [-0.52952657 -0.09016299 4.58223308 1.21708383 0.77979151
  -0.62206428]
 [-0.64586341 0.87544396 2.45763672 2.04282146 1.52094435
  -0.20672466]
 [0.13546632 -0.9126843 -1.2858428 -1.50963355 -0.86554784
  -0.5533044]
 [1.29939839 2.20837368 -10.84697168 -2.55568587 -2.00907349
   1.98296489]]
ws1_when_highest_train_accuracy: [-3.7045848   4.01637569 2.77005027 -3.47517339
-11.62935917]

Epoch_when_highest_test_accuracy: 65
Train_accuracy_when_highest_test_accuracy: 0.986245855336
Test_accuracy_when_highest_test_accuracy: 0.962264150943
ws0_when_highest_test_accuracy: [[0.06973108 -0.53672257 -1.8176734 -1.71192776
-0.83855236
  -0.27647565]
 [-0.5069374 -0.02620843 4.39911593 1.13929303 0.68364575
  -0.65511459]
 [-0.49578932 0.72502931 2.40966697 1.81113998 1.3176782
  -0.24142692]
 [-0.09544615 -0.81443741 -1.24990808 -1.28860936 -0.71797436
  -0.6045179]
 [1.55777042 1.8536126 -10.42410292 -2.64547018 -1.72267215
   1.93720771]]
ws1_when_highest_test_accuracy: [-3.37045777 3.95415998 2.62382891 -3.18345804
-11.12386093]

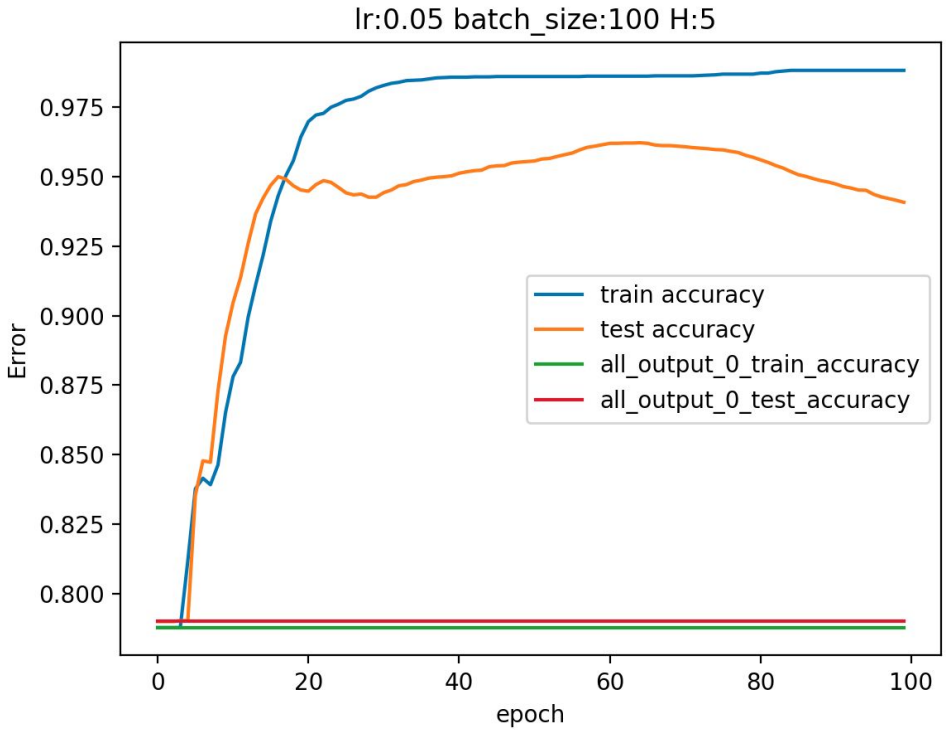

Figure 3b.1

Setting: learning rate:0.05 batch size:100 H:5 (when highest test accuracy)

	Part 2 (2e) excluding time	Part 3 including time
Train Accuracy	**98.50%**	**98.62%**
Test Accuracy	**99.26%**	**96.21%**

Table 3b.1

Code:

Environment:
python 2.7.10

Repo:

https://github.com/BryanBo-Cao/neuralnets-deeplearning Note that this repo is private for now, once this course ends it will be public. Please ask me to add you as collaborative to see these code if you would like to.

ToRun:

Open a terminal, cd to "assignment2-BoCao/src/python" directory, run **python part*.py** directly. For instance, if you want to run **part1.py** code, type **python part1.py**.

3 parts solutions are **part1*.py**, **part2*.py** and **part3*.py** with each settings in the names respectively. For instance, **part2_lrdot001_H5.py** means part2 code in the setting of learning rate as 0.001 and the number of hidden layer units is 5.

I've put a lot of efforts in this assignment, each part's code was written from scratch. You may see lots of commits from my github repo.