Xie Zhou 912143385

ECS171 HW2

Note:

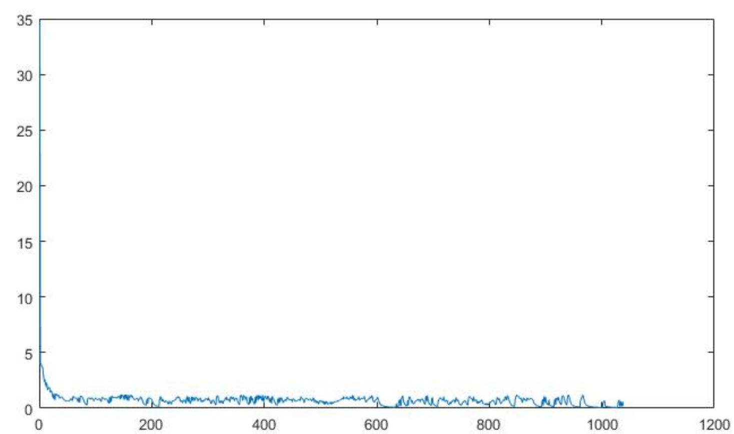
1) Initial weights are randomized in ann.m.
2) All error's are calculated using abs(A-B) instead of squared error.
3) --inputdata.m reads data from source file
   --converts information to matrix form,
   --gets rid of 1st column(useless information)
   --and outputs 1484*9 matrix.
   --Last column contains numbers instead of class numbers.

Problem1:

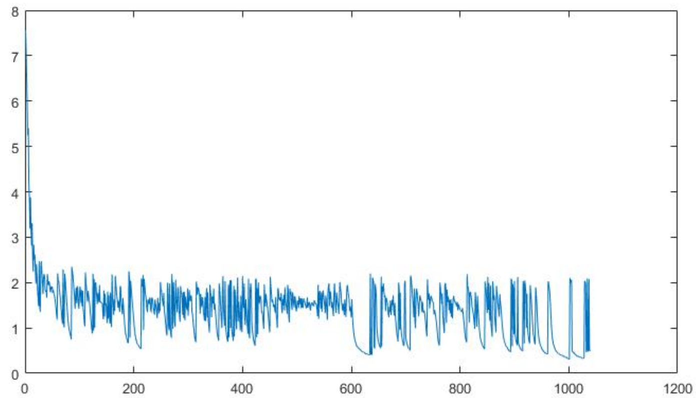
files: inputdata.m   ann.m   hidden1node3.m;
run command: hidden1node3

Answers:

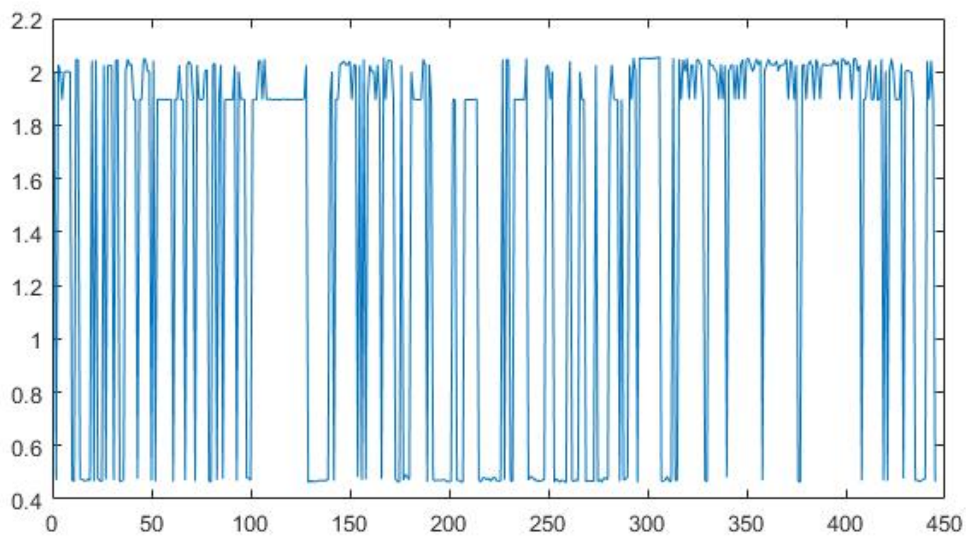
1) plot: training set weight change
   x-axis—iteration from sample1 to 1039
   y-axis—sum of weight change at each iteration



2) plot: training set error change
   x-axis—iteration from sample1 to 1039
   y-axis—sum of weight change at each iteration



3) testing set prediction outputs
   see variable "resultMatrix" in running result. Each row is a sample output
4) plot: testing error
   x-axis—testing sample from 1040 to 1484 (1 to 445)
   y-axis—testing error (sum of abs(y-prediction)), y is a binary vector



Problem2:
<u>files:</u> inputdata.m ann.m p2.m;
<u>run command</u>: p2

Answers:
1) Training error: 2.2316e+3
   Training accuracy: 31.20%
2) final activation functions:
   layer2 w_0 = 1/(1-e^(-(0.5799 + 0.2050*x1 + 0.0185*x2 + 0.7567*x3 + 0.2863*x4 + 0.2050*x5 + 0.7936*x6 + 0.8179*x7 + 0.2333*x8)))

layer2 a_1 = 1/(1-e^(-(2.4306 + 1.3397*x1 + 1.4137*x2 + 1.3028*x3 + 1.2038*x4 + 1.2391*x5 + 0.8369*x6 + 1.5090*x7 + 0.6184*x8)))

layer2 a_2 = 1/(1-e^(-(0.7457 + 0.2568*x1 + 1.0288*x2 + 0.3887*x3 + 0.2940*x4 + 0.2364*x5 + 0.9021*x6 + 0.1357*x7 + 0.2316*x8)))

layer2 a_3 = 1/(1-e^(-(1.9900 + 1.0449*x1 + 1.4448*x2 + 0.8066*x3 + 1.0059*x4 + 0.9440*x5 + 0.5672*x6 + 0.7562*x7 + 0.2590*x8)))

layer3 a_1 = 1/(1-e^(-(-0.3705*w0 - 0.1689*a_1(layer2) - 0.4609*a_2(layer2) - 0.2373*a_3(layer2))))

layer3 a_2 = 1/(1-e^(-(0.0151*w0 - 0.3841*a_1(layer2) + 0.0524*a_2(layer2) - 0.9907*a_3(layer2))))

layer3 a_3 = 1/(1-e^(-(-0.8521*w0 - 0.8275*a_1(layer2) - 0.3918*a_2(layer2) - 0.0651*a_3(layer2))))

layer3 a_4 = 1/(1-e^(-(-0.8708*w0 - 0.4023*a_1(layer2) - 0.5735*a_2(layer2) - 0.6999*a_3(layer2))))

layer3 a_5 = 1/(1-e^(-(-1.3159*w0 - 1.0144*a_1(layer2) - 0.7337*a_2(layer2) - 0.6985*a_3(layer2))))

layer3 a_6 = 1/(1-e^(-(-1.0290*w0 - 1.0607*a_1(layer2) - 0.6715*a_2(layer2) - 0.2691*a_3(layer2))))

layer3 a_7 = 1/(1-e^(-(-1.0740*w0 - 0.6608*a_1(layer2) - 1.6932*a_2(layer2) - 0.6491*a_3(layer2))))

layer3 a_8 = 1/(1-e^(-(-1.0353*w0 - 0.9082*a_1(layer2) - 0.8234*a_2(layer2) - 1.1014*a_3(layer2))))

layer3 a_9 = 1/(1-e^(-(-1.7476*w0 - 0.9713*a_1(layer2) - 0.9509*a_2(layer2) - 0.7257*a_3(layer2))))

layer3 a_10 =1/(1-e^(-(-1.4387*w0 - 1.0934*a_1(layer2) - 1.1992*a_2(layer2) - 1.0989*a_3(layer2))))

Note: w0 here represention w0*x0 where x0 is the constant 1

Problem3:

files: inputdata.m ann.m p3.m;

run command: p3

files for reference: initial_weights_first_round_weights

Answers:

1) reference file has: initial weights, initial a's, first round weights.
   Hand prove: use initial a and w to back propagate and get updated weights
   Compare updated weights and first_round_weights (1st iteration of sample 1):

   For the 2nd layer weights, hand calculation and computer results are extremely similar

   For the 1st layer weights, hand calculation and computer results differ by a very small amount. It is possibly due to rounding error since I didn't use full floating-point values for my calculation.

Layer 1

$$\frac{\partial RSS}{w_{ji}^{1}} = -\sum \delta_k^3 \cdot w_{kj}^2 \cdot a_j^2 (1-a_j^2) \cdot a_i^1$$

Layer 2

$$w_{kj}^2 = w_{kj}^2 - 1 \cdot \frac{\partial RSS}{\partial w_{kj}^2}$$

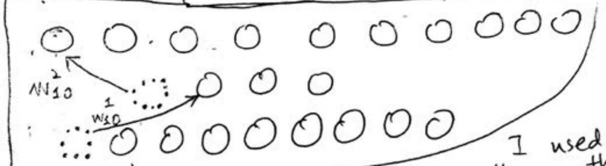
$$\frac{\partial RSS}{\partial w_{kj}^2} = -\delta_k^3 \cdot a_j^2$$

$$\delta_k^3 = (y_k - a_k^3) \cdot a_k^3 \cdot (1-a_k^3)$$

$y = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

3^{rd}

Note: layer 1 first row of w is never updated because $W \cdot X_0$ is not involved in activation functions



I used calculator to do the calculation

layer 2

$$w_{10}^2 = w_{10}^2 + 1 \cdot (y_1 - a_1^3) \cdot a_1^3 (1-a_1^3) \cdot a_0^2 = 0.3902$$

$$w_{11}^2 = w_{11}^2 + 1 \cdot (y_1 - a_1^3) \cdot a_1^3 (1-a_1^3) \cdot a_1^2 = 0.4350$$

$$w_{12}^2 = w_{12}^2 + 1 (y_1 - a_1^3) \cdot a_1^3 (1-a_1^3) \cdot a_2^2 = 0.7127$$

$$w_{13}^2 = w_{13}^2 + 1 (y_1 - a_1^3) \cdot a_1^3 (1-a_1^3) \cdot a_3^3 = 0.9144$$

computer

$$w_{20}^2 = w_{20}^2 + 1 \cdot (y_2 - a_2^3) \cdot a_2^3 (1-a_2^3) \cdot a_0^2 = 0.5230 \sim 0.5231$$

$$w_{21}^2 = w_{21}^2 + 1 \cdot (y_2 - a_2^3) \cdot a_2^3 (1-a_2^3) \cdot a_1^2 = 0.0030 \sim 0.0031$$

- - - - - - - -

$$w_{30}^2 = w_{30}^2 + 1 (y_3 - a_3^3) \cdot a_3^3 (1-a_3^3) \cdot a_0^2 = 1.0158$$

1

- - - - - - - -

$$w_{10,0}^2 = w_{10,0}^2 + (y_{10} - a_{10}^3) \cdot a_{10}^3 (1-a_{10}^3) \cdot a_0^2 = 0.5952$$

computer

layer 1:

$$w_{10}^1 = w_{10}^1 + \sum (y_k^3 - a_k^3) \cdot a_k^3 \cdot (1-a_k^3) \cdot w_{k1}^2 \cdot a_1^2 (1-a_1^2) \cdot a_0^1 = 0.7458 \sim 0.7566$$

$$w_{11}^1 = w_{11}^1 + \sum (y_k^3 - a_k^3) \cdot a_k^3 (1-a_k^3) w_{k1}^2 \cdot a_1^2 (1-a_1^2) a_1^1 = -0.0219 \sim -0.0157$$

$$w_{12}^1 = w_{12}^1 + \cdots a_2^1 = 0.3899 \sim 0.3965$$

- - - - - - - -

$$w_{34}^1 = w_{34}^1 + \sum (y_k^3 - a_k^3) \cdot a_k^3 (1-a_k^3) \cdot w_{k3}^2 \cdot a_3^2 (1-a_3^2) a_4^1 = 0.3228 \sim 0.3237$$

- - - - - - - -

$$w_{38}^1 = w_{38}^1 + \cdots a_8^1 = 0.9177 \sim 0.9191$$

Problem4:
files: inputdata.m ann.m p4.m;
run command: p4

Answers:
Call function ann recursively to get 3*4 testing set error matrix

1) Testing set error matrix:

780.5943	750.2627	733.4707	1005.679
777.2998	740.4118	728.7110	1676.981
782.5728	739.3853	892.8145	2780.879

---After running the code for several times and increasing # nodes per hidden layer/# hidden layers, (I tried 10*10 matrix) I observed that:
a) Larger #nodes per hidden layer, larger error
b) Larger #hidden layers, slightly larger errors (true when learningRate > 1)
c) Larger #hidden layers, less variation for each prediction output (looking at "resultMatrix" for 445 testing samples)

---I think observations could different (especially for "a)") if we had a larger sample size so that the program could learning better; increasing # hidden layers largely increases complexity of the model (needs more input information).

---Testing accuracy behaves similar to error, being around 0.3281 for most attribute combinations.

2) For our attribute combinations, I picked 3 nodesPerLayer, 3 hiddenLayers for optimal configuration (for prediction in next problem).

Problem5:
files: inputdata.m ann.m p5.m;
run command: p5

Answers:
(use 3 nodesPerHiddenLayer, 3 hiddenLayers)
The unknown sample was classified to:
class 1 for most of the time,
class 2 sometimes since the probability is similar to class 1,
other classes very rarely.

--Prediction output shown after running the code.

Problem6:
files: inputdata.m   p6.m;
run command: p6

Answer:
---For uncertainty, I defined a value called “variation”.
---First, calculate the means for each feature by averaging the inputs from 1484 samples, calling it “feature_avg”.
---Then, given an unknown sample, I took the abs(feature_avg [i]–sample_feature[i]), and then took the sum of the absolute values for 8 features. I called this value “variation”---how far away is the unknown input from the average sample inputs.
---Then for the uncertainty, I divided the variation by the sum of feature_avg, and multiply it by 100 to get the percent uncertainty.

For the previous unknown sample:
Variation(uncertainty):   0.273039
Uncertainty(percent uncertainty):   %8.953347