CSC 578 Project 1

Yiyang Yang

Completed:

Pseudocode ×

Well-documented ×

Code works ×

Learning parameters:

inputs, targets, nodeLayers, learning rate, ×

epochs, batchSize ×

Termination conditions: max epochs, correct classification ×

Mini-batch SGD ×

Outputs as requested (iris, MNIST, and XOR) ×

Incomplete, Details \_\_\_\_\_\_×\_\_\_\_\_\_\_

Not sure, Details \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pseudocode:**

*Initialize weights (weight) and biases (bias)*

*Starting from the second layer, for each layer*

*Initialize weight and bias, with normal distributed value.*

*Initialize batches (b), target (targetB) and a tracker (n)*

*For each batch (b)*

*If actual batch size < batch size*

*Insert actual size into b and related target into the target.*

*Else*

*Insert batch into the b, related target into the targeB and add 1 to n.*

*For each epoch (p)*

*For each batch (q)*

*Initialize cell (z) and cell (activation) to store values and activations*

*Calculate intermediate value for each layer and store it.*

*Calculate error (e) ,sigmoid gradient (sigP) and cost = e \* sigP.*

*According to backpropagation, get delta (delta) of each layer from second one*

*= weight \* delta \* sigP*

*Update weight (newW) and bias(newB)*

*newW = weight - eta / number of observation \* delta \* activation*

*newB = bias – eta / number of observation \* sum of delta.*

*Initial result (result) cell*

*First element of result is inputs.*

*Calculate MSE, correct cases and accuracy rate.*

*Ouput the result*

**Instructions:**

The neural network I create is in the file Yiyang\_NNet.m, the parameters are the same as the project document requirement. (Copy from document requirement)

* inputs: a matrix with a column for each example, and a row for each input feature
* targets: a matrix with a column for each example, and a row for each output feature
* nodeLayers: a vector with the number of nodes in each layer (including the input and output layers). **Important**: Your code should not assume that there are just three layers of nodes. It should work with a network of any size.
* numEpochs: (scalar) desired number of epochs to run
* batchSize: (scalar) number of instances in a mini-batch
* eta: (scalar) learning rate

I apply my network to the given three datasets iris.csv, xor.csv, mnisttrn.mat and get the message.

**Description:**

I follow the description of each step in the NNDL and each part of python code in NNDL, find the logical relations of each parameters and convert it into Matlab on my own.

I

**Code:**

%Data: 10/06/2017

%Author: Yiyang Yang

function [weight, bias] = Yiyang\_NNet(inputs, targets, nodeLayers, numEpochs, batchSize, eta)

%Initialize weight and bias

for i = 2: length(nodeLayers)

weight{i} = normrnd(0,1,nodeLayers(i), nodeLayers(i - 1));

bias{i} = normrnd(0,1,nodeLayers(i), 1);

end

%Batching

b = {};

targetB = {};

n = 1;

for i =1 : batchSize : length(inputs)

if length(inputs) - i < batchSize

miniB = inputs(:, i : end);

b{n} = miniB;

target = targets(:, i: end);

targetB{n} = target;

else

miniB = inputs(:, i: i + batchSize - 1);

b{n} = miniB;

target = targets(:, i: i + batchSize - 1);

targetB{n} = target;

n = n + 1;

end

end

%Epochs

for p = 1 : numEpochs

for q = 1 : length(b)

z = {};

activation = {};

activation{1} = b{q};

delta = {};

for l = 2 : length(nodeLayers)

z{l} = bsxfun(@plus, (weight{l} \* activation{l - 1}), bias{l});

activation{l} = logsig(z{l});

end

%Calculate error

lg = logsig(z{length(nodeLayers)});

sigP = lg .\* (1 - lg);

e = (activation{length(nodeLayers)} - targetB{q});

delta{length(nodeLayers)} = sigP .\* e;

%Apply Backpropagation

for l = (length(nodeLayers) - 1) : -1 : 2

delta{l} = (weight{l + 1}.' \* delta{l + 1}) .\* (logsig(z{l}) .\* (1 - logsig(z{l})));

end

%Apple gradient desent and get new weight and bias

for l = length(nodeLayers) : -1 : 2

newW = weight{l} - eta / length(b{q}) \* delta{l} \* activation{l - 1}.';

weight{l} = newW;

newB = bias{l} - eta / length(b{q}) \* sum(delta{l}, 2);

bias{l} = newB;

end

end

%Get result and ouput the message

result = {};

result{1} = inputs;

for l = 2 : length(nodeLayers)

z = bsxfun(@plus, (weight{l} \* result{l - 1}), bias{l});

result{l} = logsig(z);

end

%Get MSE, correct cases and accuracy rate

MSE = sum(sum((0.5 \* (targets - result{length(nodeLayers)}).^2))) \* 1/(2\*(length(inputs)));

if size(targets, 1) <= 1

cArray = target - round(result{length(nodeLayers)});

cNum = sum(cArray(:) == 0);

acc = cNum / length(inputs);

else

[i, v] = max(result{length(nodeLayers)});

[it, vt] = max(targets);

cArray = vt - v;

cNum = sum(cArray(:) == 0);

acc = cNum / length(inputs);

end

output = 'Epoch %d, MSE: %f, Correct: %d/%d, Acc: %f \n';

fprintf(output, p, MSE, cNum, length(targets), acc);

end

**Outputs:**

IRIS DATASET:

>> iris = csvread('D:\CSC 578\iris.csv');

>> inputs = iris(:, 1: 4).';

>> targets = iris(:, 5 : 7).';

>> Yiyang\_NNet(inputs, targets, [4, 20, 3], 100, 10, 0.1);

Epoch 1, MSE: 0.374738, Correct: 50/150, Acc: 0.333333

Epoch 2, MSE: 0.323282, Correct: 50/150, Acc: 0.333333

Epoch 3, MSE: 0.283147, Correct: 50/150, Acc: 0.333333

Epoch 4, MSE: 0.209643, Correct: 50/150, Acc: 0.333333

Epoch 5, MSE: 0.177899, Correct: 50/150, Acc: 0.333333

Epoch 6, MSE: 0.162143, Correct: 50/150, Acc: 0.333333

Epoch 7, MSE: 0.146050, Correct: 52/150, Acc: 0.346667

Epoch 8, MSE: 0.133122, Correct: 99/150, Acc: 0.660000

Epoch 9, MSE: 0.124909, Correct: 100/150, Acc: 0.666667

Epoch 10, MSE: 0.119388, Correct: 100/150, Acc: 0.666667

Epoch 11, MSE: 0.115341, Correct: 100/150, Acc: 0.666667

Epoch 12, MSE: 0.112198, Correct: 100/150, Acc: 0.666667

Epoch 13, MSE: 0.109652, Correct: 100/150, Acc: 0.666667

Epoch 14, MSE: 0.107508, Correct: 100/150, Acc: 0.666667

Epoch 15, MSE: 0.105645, Correct: 100/150, Acc: 0.666667

Epoch 16, MSE: 0.103981, Correct: 100/150, Acc: 0.666667

Epoch 17, MSE: 0.102463, Correct: 100/150, Acc: 0.666667

Epoch 18, MSE: 0.101052, Correct: 100/150, Acc: 0.666667

Epoch 19, MSE: 0.099719, Correct: 100/150, Acc: 0.666667

Epoch 20, MSE: 0.098439, Correct: 100/150, Acc: 0.666667

Epoch 21, MSE: 0.097192, Correct: 100/150, Acc: 0.666667

Epoch 22, MSE: 0.095962, Correct: 100/150, Acc: 0.666667

Epoch 23, MSE: 0.094738, Correct: 100/150, Acc: 0.666667

Epoch 24, MSE: 0.093523, Correct: 100/150, Acc: 0.666667

Epoch 25, MSE: 0.092335, Correct: 100/150, Acc: 0.666667

Epoch 26, MSE: 0.091206, Correct: 100/150, Acc: 0.666667

Epoch 27, MSE: 0.090164, Correct: 100/150, Acc: 0.666667

Epoch 28, MSE: 0.089223, Correct: 100/150, Acc: 0.666667

Epoch 29, MSE: 0.088381, Correct: 100/150, Acc: 0.666667

Epoch 30, MSE: 0.087626, Correct: 100/150, Acc: 0.666667

Epoch 31, MSE: 0.086939, Correct: 100/150, Acc: 0.666667

Epoch 32, MSE: 0.086306, Correct: 100/150, Acc: 0.666667

Epoch 33, MSE: 0.085715, Correct: 100/150, Acc: 0.666667

Epoch 34, MSE: 0.085154, Correct: 100/150, Acc: 0.666667

Epoch 35, MSE: 0.084619, Correct: 100/150, Acc: 0.666667

Epoch 36, MSE: 0.084102, Correct: 101/150, Acc: 0.673333

Epoch 37, MSE: 0.083601, Correct: 101/150, Acc: 0.673333

Epoch 38, MSE: 0.083113, Correct: 101/150, Acc: 0.673333

Epoch 39, MSE: 0.082635, Correct: 101/150, Acc: 0.673333

Epoch 40, MSE: 0.082166, Correct: 101/150, Acc: 0.673333

Epoch 41, MSE: 0.081706, Correct: 102/150, Acc: 0.680000

Epoch 42, MSE: 0.081252, Correct: 102/150, Acc: 0.680000

Epoch 43, MSE: 0.080804, Correct: 102/150, Acc: 0.680000

Epoch 44, MSE: 0.080362, Correct: 102/150, Acc: 0.680000

Epoch 45, MSE: 0.079924, Correct: 102/150, Acc: 0.680000

Epoch 46, MSE: 0.079491, Correct: 102/150, Acc: 0.680000

Epoch 47, MSE: 0.079062, Correct: 102/150, Acc: 0.680000

Epoch 48, MSE: 0.078637, Correct: 102/150, Acc: 0.680000

Epoch 49, MSE: 0.078215, Correct: 102/150, Acc: 0.680000

Epoch 50, MSE: 0.077796, Correct: 102/150, Acc: 0.680000

Epoch 51, MSE: 0.077380, Correct: 102/150, Acc: 0.680000

Epoch 52, MSE: 0.076967, Correct: 102/150, Acc: 0.680000

Epoch 53, MSE: 0.076556, Correct: 103/150, Acc: 0.686667

Epoch 54, MSE: 0.076147, Correct: 103/150, Acc: 0.686667

Epoch 55, MSE: 0.075740, Correct: 104/150, Acc: 0.693333

Epoch 56, MSE: 0.075335, Correct: 105/150, Acc: 0.700000

Epoch 57, MSE: 0.074931, Correct: 105/150, Acc: 0.700000

Epoch 58, MSE: 0.074529, Correct: 105/150, Acc: 0.700000

Epoch 59, MSE: 0.074128, Correct: 105/150, Acc: 0.700000

Epoch 60, MSE: 0.073729, Correct: 106/150, Acc: 0.706667

Epoch 61, MSE: 0.073330, Correct: 106/150, Acc: 0.706667

Epoch 62, MSE: 0.072933, Correct: 106/150, Acc: 0.706667

Epoch 63, MSE: 0.072536, Correct: 106/150, Acc: 0.706667

Epoch 64, MSE: 0.072141, Correct: 107/150, Acc: 0.713333

Epoch 65, MSE: 0.071746, Correct: 108/150, Acc: 0.720000

Epoch 66, MSE: 0.071352, Correct: 108/150, Acc: 0.720000

Epoch 67, MSE: 0.070958, Correct: 110/150, Acc: 0.733333

Epoch 68, MSE: 0.070565, Correct: 110/150, Acc: 0.733333

Epoch 69, MSE: 0.070173, Correct: 110/150, Acc: 0.733333

Epoch 70, MSE: 0.069781, Correct: 110/150, Acc: 0.733333

Epoch 71, MSE: 0.069390, Correct: 110/150, Acc: 0.733333

Epoch 72, MSE: 0.068999, Correct: 112/150, Acc: 0.746667

Epoch 73, MSE: 0.068608, Correct: 112/150, Acc: 0.746667

Epoch 74, MSE: 0.068218, Correct: 114/150, Acc: 0.760000

Epoch 75, MSE: 0.067829, Correct: 115/150, Acc: 0.766667

Epoch 76, MSE: 0.067439, Correct: 115/150, Acc: 0.766667

Epoch 77, MSE: 0.067051, Correct: 116/150, Acc: 0.773333

Epoch 78, MSE: 0.066662, Correct: 116/150, Acc: 0.773333

Epoch 79, MSE: 0.066274, Correct: 118/150, Acc: 0.786667

Epoch 80, MSE: 0.065886, Correct: 118/150, Acc: 0.786667

Epoch 81, MSE: 0.065499, Correct: 118/150, Acc: 0.786667

Epoch 82, MSE: 0.065112, Correct: 118/150, Acc: 0.786667

Epoch 83, MSE: 0.064726, Correct: 119/150, Acc: 0.793333

Epoch 84, MSE: 0.064340, Correct: 119/150, Acc: 0.793333

Epoch 85, MSE: 0.063955, Correct: 119/150, Acc: 0.793333

Epoch 86, MSE: 0.063571, Correct: 119/150, Acc: 0.793333

Epoch 87, MSE: 0.063187, Correct: 119/150, Acc: 0.793333

Epoch 88, MSE: 0.062803, Correct: 119/150, Acc: 0.793333

Epoch 89, MSE: 0.062420, Correct: 120/150, Acc: 0.800000

Epoch 90, MSE: 0.062038, Correct: 120/150, Acc: 0.800000

Epoch 91, MSE: 0.061657, Correct: 120/150, Acc: 0.800000

Epoch 92, MSE: 0.061277, Correct: 120/150, Acc: 0.800000

Epoch 93, MSE: 0.060897, Correct: 120/150, Acc: 0.800000

Epoch 94, MSE: 0.060519, Correct: 120/150, Acc: 0.800000

Epoch 95, MSE: 0.060141, Correct: 121/150, Acc: 0.806667

Epoch 96, MSE: 0.059764, Correct: 121/150, Acc: 0.806667

Epoch 97, MSE: 0.059389, Correct: 121/150, Acc: 0.806667

Epoch 98, MSE: 0.059014, Correct: 122/150, Acc: 0.813333

Epoch 99, MSE: 0.058641, Correct: 122/150, Acc: 0.813333

Epoch 100, MSE: 0.058269, Correct: 123/150, Acc: 0.820000

MNIST DATASET

>> load('D:\CSC 578\mnistTrn.mat');

>> Yiyang\_NNet(trn, trnAns, [784, 30, 10], 30, 10, 3);

Epoch 1, MSE: 0.239845, Correct: 8145/50000, Acc: 0.162900

Epoch 2, MSE: 0.204009, Correct: 17255/50000, Acc: 0.345100

Epoch 3, MSE: 0.172070, Correct: 23780/50000, Acc: 0.475600

Epoch 4, MSE: 0.147844, Correct: 28172/50000, Acc: 0.563440

Epoch 5, MSE: 0.135889, Correct: 29992/50000, Acc: 0.599840

Epoch 6, MSE: 0.125793, Correct: 32061/50000, Acc: 0.641220

Epoch 7, MSE: 0.114534, Correct: 34154/50000, Acc: 0.683080

Epoch 8, MSE: 0.105919, Correct: 35453/50000, Acc: 0.709060

Epoch 9, MSE: 0.099825, Correct: 36266/50000, Acc: 0.725320

Epoch 10, MSE: 0.095165, Correct: 36885/50000, Acc: 0.737700

Epoch 11, MSE: 0.091375, Correct: 37339/50000, Acc: 0.746780

Epoch 12, MSE: 0.088188, Correct: 37768/50000, Acc: 0.755360

Epoch 13, MSE: 0.085365, Correct: 38186/50000, Acc: 0.763720

Epoch 14, MSE: 0.082344, Correct: 38779/50000, Acc: 0.775580

Epoch 15, MSE: 0.077716, Correct: 40226/50000, Acc: 0.804520

Epoch 16, MSE: 0.072784, Correct: 41160/50000, Acc: 0.823200

Epoch 17, MSE: 0.069090, Correct: 41667/50000, Acc: 0.833340

Epoch 18, MSE: 0.066284, Correct: 42005/50000, Acc: 0.840100

Epoch 19, MSE: 0.064004, Correct: 42298/50000, Acc: 0.845960

Epoch 20, MSE: 0.062066, Correct: 42537/50000, Acc: 0.850740

Epoch 21, MSE: 0.060373, Correct: 42740/50000, Acc: 0.854800

Epoch 22, MSE: 0.058869, Correct: 42908/50000, Acc: 0.858160

Epoch 23, MSE: 0.057517, Correct: 43109/50000, Acc: 0.862180

Epoch 24, MSE: 0.056289, Correct: 43256/50000, Acc: 0.865120

Epoch 25, MSE: 0.055164, Correct: 43400/50000, Acc: 0.868000

Epoch 26, MSE: 0.054127, Correct: 43517/50000, Acc: 0.870340

Epoch 27, MSE: 0.053166, Correct: 43616/50000, Acc: 0.872320

Epoch 28, MSE: 0.052269, Correct: 43714/50000, Acc: 0.874280

Epoch 29, MSE: 0.051430, Correct: 43824/50000, Acc: 0.876480

Epoch 30, MSE: 0.050641, Correct: 43914/50000, Acc: 0.878280

XOR DATASET:

>> input = [0 0 1 1; 0 1 0 1];

>> target = [0 1 1 0];

>> Yiyang\_NNet(input, target, [2, 2, 1], 10, 4, 0.1);

Epoch 1, MSE: 0.061687, Correct: 1/4, Acc: 0.250000

Epoch 2, MSE: 0.061683, Correct: 1/4, Acc: 0.250000

Epoch 3, MSE: 0.061679, Correct: 1/4, Acc: 0.250000

Epoch 4, MSE: 0.061675, Correct: 1/4, Acc: 0.250000

Epoch 5, MSE: 0.061671, Correct: 1/4, Acc: 0.250000

Epoch 6, MSE: 0.061667, Correct: 1/4, Acc: 0.250000

Epoch 7, MSE: 0.061663, Correct: 1/4, Acc: 0.250000

Epoch 8, MSE: 0.061660, Correct: 1/4, Acc: 0.250000

Epoch 9, MSE: 0.061656, Correct: 1/4, Acc: 0.250000

Epoch 10, MSE: 0.061653, Correct: 1/4, Acc: 0.250000

>> Yiyang\_NNet(input, target, [2, 2, 1], 10, 1, 0.1);

Epoch 1, MSE: 0.084211, Correct: 4/4, Acc: 1.000000

Epoch 2, MSE: 0.083714, Correct: 4/4, Acc: 1.000000

Epoch 3, MSE: 0.083216, Correct: 4/4, Acc: 1.000000

Epoch 4, MSE: 0.082718, Correct: 4/4, Acc: 1.000000

Epoch 5, MSE: 0.082220, Correct: 4/4, Acc: 1.000000

Epoch 6, MSE: 0.081724, Correct: 4/4, Acc: 1.000000

Epoch 7, MSE: 0.081229, Correct: 4/4, Acc: 1.000000

Epoch 8, MSE: 0.080736, Correct: 4/4, Acc: 1.000000

Epoch 9, MSE: 0.080246, Correct: 4/4, Acc: 1.000000

Epoch 10, MSE: 0.079758, Correct: 4/4, Acc: 1.000000

>> Yiyang\_NNet(input, target, [2, 3, 2, 1], 20, 1, 0.1);

Epoch 1, MSE: 0.099793, Correct: 4/4, Acc: 1.000000

Epoch 2, MSE: 0.099502, Correct: 4/4, Acc: 1.000000

Epoch 3, MSE: 0.099208, Correct: 4/4, Acc: 1.000000

Epoch 4, MSE: 0.098909, Correct: 4/4, Acc: 1.000000

Epoch 5, MSE: 0.098606, Correct: 4/4, Acc: 1.000000

Epoch 6, MSE: 0.098299, Correct: 4/4, Acc: 1.000000

Epoch 7, MSE: 0.097988, Correct: 4/4, Acc: 1.000000

Epoch 8, MSE: 0.097672, Correct: 4/4, Acc: 1.000000

Epoch 9, MSE: 0.097352, Correct: 4/4, Acc: 1.000000

Epoch 10, MSE: 0.097027, Correct: 4/4, Acc: 1.000000

Epoch 11, MSE: 0.096698, Correct: 4/4, Acc: 1.000000

Epoch 12, MSE: 0.096365, Correct: 4/4, Acc: 1.000000

Epoch 13, MSE: 0.096028, Correct: 4/4, Acc: 1.000000

Epoch 14, MSE: 0.095686, Correct: 4/4, Acc: 1.000000

Epoch 15, MSE: 0.095340, Correct: 4/4, Acc: 1.000000

Epoch 16, MSE: 0.094989, Correct: 4/4, Acc: 1.000000

Epoch 17, MSE: 0.094635, Correct: 4/4, Acc: 1.000000

Epoch 18, MSE: 0.094276, Correct: 4/4, Acc: 1.000000

Epoch 19, MSE: 0.093913, Correct: 4/4, Acc: 1.000000

Epoch 20, MSE: 0.093545, Correct: 4/4, Acc: 1.000000

**Analysis:**

I find that, my network works well in iris and mnist problem, the correct case and accuracy rate is good. In the first part of XOR problem, my network is not working well, I think the reason is that the batch size is larger than what it should be, in second and third part, it works pretty good, the accuracy rate is 1.0.

**Ideas for enhancement:**

I think my network still need improvement, this is a very sample network, I think after I learn more about neural network I will know what part I should keep working on.