Two-layer perceptron (2020)

clear all

M1 = 8;

M2 = 4;

step = .02;

numRuns = 5000;

% training variables

train = readmatrix('training\_set.csv');

X = train(:,1:2);

t = train(:,3);

pVal = length(X);

w{1} = -.2 + .4.\*rand(M1,2);

w{2} = -.2 + .4.\*rand(M2,M1);

w{3} = -.2 + .4.\*rand(1,M2);

theta{1} = zeros(M1,1);

theta{2} = zeros(M2,1);

theta{3} = 0;

B{1} = zeros(M1,pVal);

B{2} = zeros(M2,pVal);

B{3} = zeros(1,pVal);

V{1} = zeros(M1,pVal);

V{2} = zeros(M2,pVal);

V{3} = zeros(1,pVal);

sigO = zeros(1,pVal);

err{1} = zeros(M1);

err{2} = zeros(M2);

err{3} = zeros(1);

C = zeros(1,numRuns);

% testing variables

test = readmatrix('validation\_set.csv');

testInput = test(:,1:2);

testTarget = test(:,3);

pValTest = length(testInput);

wGood{1} = zeros(M1,2);

wGood{2} = zeros(M2,M1);

wGood{3} = zeros(1,M2);

thetaGood{1} = theta{1};

thetaGood{2} = theta{2};

thetaGOod{3} = theta{3};

BTest{1} = B{1};

BTest{2} = B{2};

BTest{3} = B{3};

VTest{1} = V{1};

VTest{2} = V{2};

VTest{3} = V{3};

sigOTest = sigO;

CTest = zeros(1,numRuns);

numIter = 0;

for iRun=1:numRuns

% calc Bs, Vs, and final output sigO

for mu=1:pVal

B{1}(:,mu) = w{1} \* X(mu,:)' - theta{1};

V{1}(:,mu) = tanh(B{1}(:,mu));

for L=2:3

B{L}(:,mu) = w{L} \* V{L-1}(:,mu) - theta{L};

V{L}(:,mu) = tanh(B{L}(:,mu));

end

if V{3}(:,mu) >= 0

sigO(:,mu) = 1;

else

sigO(:,mu) = -1;

end

C(iRun) = C(iRun) + abs(sigO(:,mu) - t(mu));

end

C(iRun) = C(iRun)/(2\*pVal);

for numFed=1:pVal

numIter = numIter + 1;

iRand = randi(pVal,1);

% calc local fields, Vs

B{1}(:,iRand) = w{1} \* X(iRand,:)' - theta{1};

V{1}(:,iRand) = tanh(B{1}(:,iRand));

for L=2:3

B{L}(:,iRand) = w{L} \* V{L-1}(:,iRand) - theta{L};

V{L}(:,iRand) = tanh(B{L}(:,iRand));

end

% calc errors

err{3} = (t(iRand)-V{3}(:,iRand)) \* (1 - tanh(B{3}(:,iRand))^2);

for L=flip(1:2)

err{L} = w{L+1}' \* err{L+1} .\* (1 - tanh(B{L}(:,iRand)).^2);

end

% update weights and biases

w{1} = w{1} + step \* err{1} \* X(iRand,:);

w{2} = w{2} + step \* err{2} \* V{1}(:,iRand)';

w{3} = w{3} + step \* err{3} \* V{2}(:,iRand)';

theta{1} = theta{1} - step \* err{1};

theta{2} = theta{2} - step \* err{2};

theta{3} = theta{3} - step \* err{3};

end

% calc C for test data

for i=1:pValTest

BTest{1}(:,i) = w{1} \* testInput(i,:)' - theta{1};

VTest{1}(:,i) = tanh(BTest{1}(:,i));

for L=2:3

BTest{L}(:,i) = w{L} \* VTest{L-1}(:,i) - theta{L};

VTest{L}(:,i) = tanh(BTest{L}(:,i));

end

if VTest{3}(:,i) >= 0

sigOTest(:,i) = 1;

else

sigOTest(:,i) = -1;

end

CTest(iRun) = CTest(iRun) + abs(sigOTest(:,i) - testTarget(i));

end

CTest(iRun) = CTest(iRun)/(2\*pValTest);

if mod(iRun,10) == 0

plot(CTest)

drawnow

end

if CTest(iRun) < .12 % store good weights and biases

for L=1:3

wGood{L} = w{L};

thetaGood{L} = theta{L};

end

end

if CTest(iRun) < .115 % stopping criteria

break

end

end