```
%Classification Neural Network by Sacha Muller
%AERO 40041 Data-Driven Methods CW2
%Group 61

%Pseudocode

%1. Data Import
%2. Structure determination (number of layers, neurons)
%3. Initial weights
%4. Input activation functions
%5. Implement forward propagation
%6. Implement cost function
%7. Implement backwards propagation
%8. Implement training function (running forward and back propagation)

%% Data Import
%import csv
clear
trainingdata = readmatrix('classification_training.csv');

%cut out NaN, don't run this codeblock without re-import csv or you will
%delete data
trainingdata(1,:) = [];
feature1 = trainingdata(:,[1 2]);
feature2 = trainingdata(:, [1 3]);
trainingclass= trainingdata(:, [1 4]);
window1 = [min(feature1(:,2)), max(feature1(:,2))];
window2 = [min(feature2(:,2)), max(feature2(:,2))];

%% Structure and Initial Weights
%constants -> change as necessary%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inputfeatures = 2;
neuronperlayer = 10;
outputneurons = 1;
layernumber = 1 ;%code can be generalized to arbitrary layer number
runhyperparamsweep = 1; %set to 0 to avoid running parameter sweep
rng(12345)

%creating weights of 1st layer
w{1} = 0.5 * rand(neuronperlayer,inputfeatures) - 0.25;

%Weights and biases of additional layers
%I know I should pre-allocate matrices for efficiency but they're small
%so its ok
for i=1:layernumber
    b{i} = zeros(neuronperlayer,1);
end

if layernumber > 1
```

```
    for i = 2:layernumber
    w{i} = 0.5 * rand(neuronperlayer,neuronperlayer) - 0.25;
    end
end

%Weights and biases of final layer

w{layernumber + 1} = 0.5 * rand(outputneurons,neuronperlayer) - 0.25;
b{layernumber + 1} = zeros(outputneurons, 1) ;

%% Training Loop

%TRAINING LOOP CONSTANTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alpha = 0.06; %learning rate;
epsilon = 1e-15; %avoid division by 0;
epochs = 100;

for i=1:epochs
%shuffle data so that every epoch has a different dataset
    shuffler = randperm(length(trainingdata));

for j=1:length(trainingdata)
%feedforward through dataset one sample at a time
samplex = [feature1(shuffler(j),2); feature2(shuffler(j),2)];
sampley = [trainingclass(shuffler(j),2)];

[n, a] = forward(samplex, w, b);

%feedback

[w, b] = backward(samplex, sampley, n, a, w, b, alpha, layernumber) ;

%evaluate cost
cost(j) = -(sampley * log(a{end} + epsilon) + (1 - sampley) * log(1 - a{end} +
epsilon));

%cost check every few hundred iterations

if mod(j,100) == 0
    cumulcost = sum(cost)/j;
    disp(['the cost at iteration ' num2str(j) ' is ' num2str(cumulcost)])
end

end
end
```

```
the cost at iteration 100 is 0.64125
the cost at iteration 200 is 0.5287
the cost at iteration 300 is 0.48019
the cost at iteration 100 is 1.0848
the cost at iteration 200 is 0.48694
```

```
the cost at iteration 300 is 0.30061
the cost at iteration 100 is 0.91998
the cost at iteration 200 is 0.4325
the cost at iteration 300 is 0.28005
the cost at iteration 100 is 0.80719
the cost at iteration 200 is 0.44738
the cost at iteration 300 is 0.27998
the cost at iteration 100 is 0.93001
the cost at iteration 200 is 0.42567
the cost at iteration 300 is 0.26871
the cost at iteration 100 is 0.73556
the cost at iteration 200 is 0.3726
the cost at iteration 300 is 0.27802
the cost at iteration 100 is 0.81057
the cost at iteration 200 is 0.37528
the cost at iteration 300 is 0.28165
the cost at iteration 100 is 0.86259
the cost at iteration 200 is 0.44916
the cost at iteration 300 is 0.27723
the cost at iteration 100 is 0.83856
the cost at iteration 200 is 0.44014
the cost at iteration 300 is 0.27613
the cost at iteration 100 is 0.75741
the cost at iteration 200 is 0.4049
the cost at iteration 300 is 0.27699
the cost at iteration 100 is 0.87747
the cost at iteration 200 is 0.39512
the cost at iteration 300 is 0.26847
the cost at iteration 100 is 0.89666
the cost at iteration 200 is 0.47984
the cost at iteration 300 is 0.27561
the cost at iteration 100 is 0.75876
the cost at iteration 200 is 0.37039
the cost at iteration 300 is 0.27527
the cost at iteration 100 is 0.8425
the cost at iteration 200 is 0.3882
the cost at iteration 300 is 0.27436
the cost at iteration 100 is 0.81336
the cost at iteration 200 is 0.40505
the cost at iteration 300 is 0.26692
the cost at iteration 100 is 0.80089
the cost at iteration 200 is 0.40523
the cost at iteration 300 is 0.26277
the cost at iteration 100 is 0.76847
the cost at iteration 200 is 0.386
the cost at iteration 300 is 0.26691
the cost at iteration 100 is 0.7117
the cost at iteration 200 is 0.37319
the cost at iteration 300 is 0.27132
the cost at iteration 100 is 0.95525
the cost at iteration 200 is 0.47403
the cost at iteration 300 is 0.26914
the cost at iteration 100 is 0.69038
the cost at iteration 200 is 0.39834
the cost at iteration 300 is 0.2747
the cost at iteration 100 is 0.99104
the cost at iteration 200 is 0.38816
the cost at iteration 300 is 0.26864
the cost at iteration 100 is 0.65361
the cost at iteration 200 is 0.40045
the cost at iteration 300 is 0.27142
the cost at iteration 100 is 0.85379
the cost at iteration 200 is 0.38074
the cost at iteration 300 is 0.26364
```

```
the cost at iteration 100 is 0.75479
the cost at iteration 200 is 0.40556
the cost at iteration 300 is 0.27084
the cost at iteration 100 is 0.84387
the cost at iteration 200 is 0.43109
the cost at iteration 300 is 0.27216
the cost at iteration 100 is 0.76199
the cost at iteration 200 is 0.40132
the cost at iteration 300 is 0.26964
the cost at iteration 100 is 0.91999
the cost at iteration 200 is 0.44778
the cost at iteration 300 is 0.26615
the cost at iteration 100 is 0.71956
the cost at iteration 200 is 0.33948
the cost at iteration 300 is 0.26929
the cost at iteration 100 is 0.8285
the cost at iteration 200 is 0.42855
the cost at iteration 300 is 0.26845
the cost at iteration 100 is 0.84208
the cost at iteration 200 is 0.37844
the cost at iteration 300 is 0.25758
the cost at iteration 100 is 0.73459
the cost at iteration 200 is 0.3835
the cost at iteration 300 is 0.26065
the cost at iteration 100 is 0.80773
the cost at iteration 200 is 0.41237
the cost at iteration 300 is 0.25517
the cost at iteration 100 is 0.71465
the cost at iteration 200 is 0.35128
the cost at iteration 300 is 0.25758
the cost at iteration 100 is 0.80376
the cost at iteration 200 is 0.40255
the cost at iteration 300 is 0.2524
the cost at iteration 100 is 0.73987
the cost at iteration 200 is 0.38628
the cost at iteration 300 is 0.25211
the cost at iteration 100 is 0.73178
the cost at iteration 200 is 0.37891
the cost at iteration 300 is 0.25411
the cost at iteration 100 is 0.73998
the cost at iteration 200 is 0.35435
the cost at iteration 300 is 0.24581
the cost at iteration 100 is 0.79541
the cost at iteration 200 is 0.40392
the cost at iteration 300 is 0.24949
the cost at iteration 100 is 0.69946
the cost at iteration 200 is 0.32656
the cost at iteration 300 is 0.24388
the cost at iteration 100 is 0.74142
the cost at iteration 200 is 0.37021
the cost at iteration 300 is 0.24512
the cost at iteration 100 is 0.74157
the cost at iteration 200 is 0.37489
the cost at iteration 300 is 0.23054
the cost at iteration 100 is 0.66688
the cost at iteration 200 is 0.33332
the cost at iteration 300 is 0.22728
the cost at iteration 100 is 0.70459
the cost at iteration 200 is 0.35551
the cost at iteration 300 is 0.21991
the cost at iteration 100 is 0.62632
the cost at iteration 200 is 0.26504
the cost at iteration 300 is 0.19478
the cost at iteration 100 is 0.5749
```

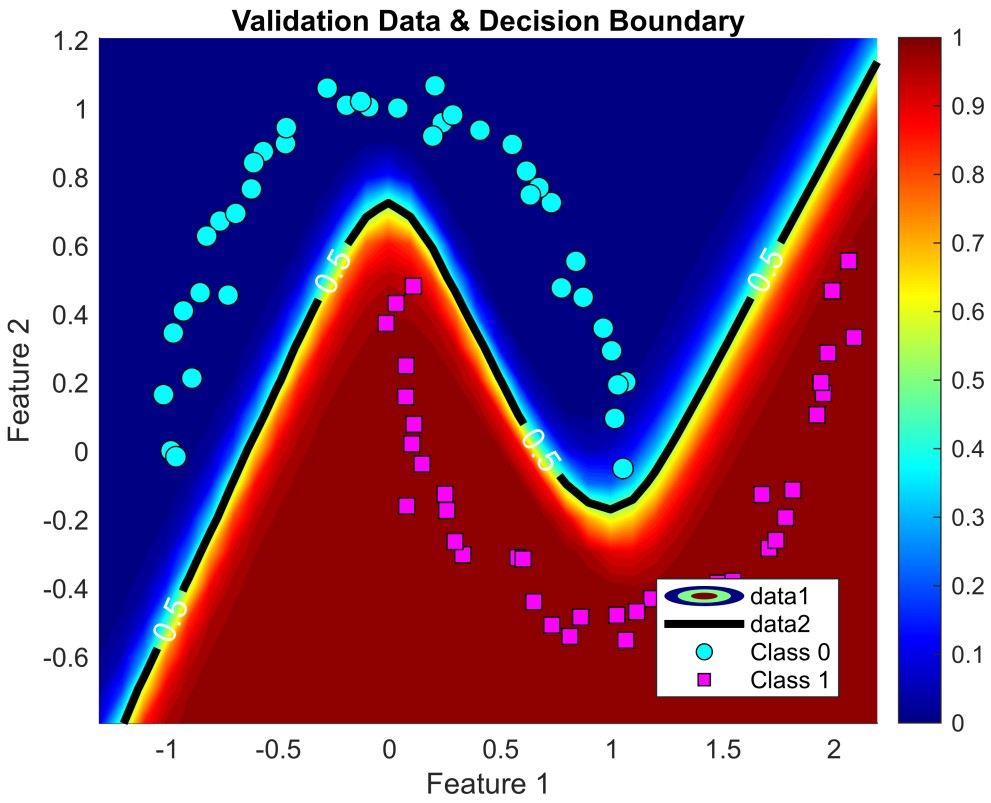
```
the cost at iteration 200 is 0.26497
the cost at iteration 300 is 0.17056
the cost at iteration 100 is 0.46564
the cost at iteration 200 is 0.24581
the cost at iteration 300 is 0.13977
the cost at iteration 100 is 0.47641
the cost at iteration 200 is 0.19646
the cost at iteration 300 is 0.11906
the cost at iteration 100 is 0.30245
the cost at iteration 200 is 0.18156
the cost at iteration 300 is 0.10703
the cost at iteration 100 is 0.27539
the cost at iteration 200 is 0.13142
the cost at iteration 300 is 0.091321
the cost at iteration 100 is 0.24587
the cost at iteration 200 is 0.11328
the cost at iteration 300 is 0.077696
the cost at iteration 100 is 0.2501
the cost at iteration 200 is 0.1148
the cost at iteration 300 is 0.070817
the cost at iteration 100 is 0.20497
the cost at iteration 200 is 0.092501
the cost at iteration 300 is 0.06272
the cost at iteration 100 is 0.17312
the cost at iteration 200 is 0.098167
the cost at iteration 300 is 0.055114
the cost at iteration 100 is 0.15374
the cost at iteration 200 is 0.07038
the cost at iteration 300 is 0.04934
the cost at iteration 100 is 0.13043
the cost at iteration 200 is 0.073038
the cost at iteration 300 is 0.043816
the cost at iteration 100 is 0.14597
the cost at iteration 200 is 0.058527
the cost at iteration 300 is 0.039784
the cost at iteration 100 is 0.099482
the cost at iteration 200 is 0.056981
the cost at iteration 300 is 0.036166
the cost at iteration 100 is 0.1163
the cost at iteration 200 is 0.052445
the cost at iteration 300 is 0.032356
the cost at iteration 100 is 0.090386
the cost at iteration 200 is 0.038337
the cost at iteration 300 is 0.029101
the cost at iteration 100 is 0.089405
the cost at iteration 200 is 0.043583
the cost at iteration 300 is 0.026061
the cost at iteration 100 is 0.087859
the cost at iteration 200 is 0.038194
the cost at iteration 300 is 0.025534
the cost at iteration 100 is 0.075499
the cost at iteration 200 is 0.040711
the cost at iteration 300 is 0.023107
the cost at iteration 100 is 0.054095
the cost at iteration 200 is 0.029358
the cost at iteration 300 is 0.020704
the cost at iteration 100 is 0.066319
the cost at iteration 200 is 0.028954
the cost at iteration 300 is 0.019842
the cost at iteration 100 is 0.05401
the cost at iteration 200 is 0.026859
the cost at iteration 300 is 0.018527
the cost at iteration 100 is 0.052773
the cost at iteration 200 is 0.023889
```

```
the cost at iteration 300 is 0.016669
the cost at iteration 100 is 0.052961
the cost at iteration 200 is 0.028922
the cost at iteration 300 is 0.016004
the cost at iteration 100 is 0.04925
the cost at iteration 200 is 0.022929
the cost at iteration 300 is 0.01549
the cost at iteration 100 is 0.043565
the cost at iteration 200 is 0.023806
the cost at iteration 300 is 0.01403
the cost at iteration 100 is 0.042076
the cost at iteration 200 is 0.018047
the cost at iteration 300 is 0.013631
the cost at iteration 100 is 0.036854
the cost at iteration 200 is 0.019251
the cost at iteration 300 is 0.012611
the cost at iteration 100 is 0.037623
the cost at iteration 200 is 0.017909
the cost at iteration 300 is 0.01236
the cost at iteration 100 is 0.039224
the cost at iteration 200 is 0.019246
the cost at iteration 300 is 0.011187
the cost at iteration 100 is 0.027049
the cost at iteration 200 is 0.014382
the cost at iteration 300 is 0.011102
the cost at iteration 100 is 0.0366
the cost at iteration 200 is 0.015926
the cost at iteration 300 is 0.010478
the cost at iteration 100 is 0.031307
the cost at iteration 200 is 0.015393
the cost at iteration 300 is 0.0097223
the cost at iteration 100 is 0.026323
the cost at iteration 200 is 0.014786
the cost at iteration 300 is 0.009686
the cost at iteration 100 is 0.028126
the cost at iteration 200 is 0.013799
the cost at iteration 300 is 0.0092966
the cost at iteration 100 is 0.026918
the cost at iteration 200 is 0.014444
the cost at iteration 300 is 0.0087664
the cost at iteration 100 is 0.02711
the cost at iteration 200 is 0.01268
the cost at iteration 300 is 0.0085241
the cost at iteration 100 is 0.025698
the cost at iteration 200 is 0.01233
the cost at iteration 300 is 0.0081285
the cost at iteration 100 is 0.022435
the cost at iteration 200 is 0.011771
the cost at iteration 300 is 0.00793
the cost at iteration 100 is 0.026657
the cost at iteration 200 is 0.012088
the cost at iteration 300 is 0.0076631
the cost at iteration 100 is 0.023759
the cost at iteration 200 is 0.010785
the cost at iteration 300 is 0.0073564
the cost at iteration 100 is 0.021326
the cost at iteration 200 is 0.011094
the cost at iteration 300 is 0.0070395
the cost at iteration 100 is 0.017195
the cost at iteration 200 is 0.0079305
the cost at iteration 300 is 0.0067932
the cost at iteration 100 is 0.023499
the cost at iteration 200 is 0.011707
the cost at iteration 300 is 0.0065775
```

```
the cost at iteration 100 is 0.018644
the cost at iteration 200 is 0.0091306
the cost at iteration 300 is 0.0064552
the cost at iteration 100 is 0.020002
the cost at iteration 200 is 0.0097678
the cost at iteration 300 is 0.0061896
the cost at iteration 100 is 0.018593
the cost at iteration 200 is 0.0099039
the cost at iteration 300 is 0.0060077
the cost at iteration 100 is 0.016244
the cost at iteration 200 is 0.0073828
the cost at iteration 300 is 0.005921
the cost at iteration 100 is 0.017967
the cost at iteration 200 is 0.0094932
the cost at iteration 300 is 0.0057013
the cost at iteration 100 is 0.015453
the cost at iteration 200 is 0.0087809
the cost at iteration 300 is 0.0055472
the cost at iteration 100 is 0.016872
the cost at iteration 200 is 0.0071463
the cost at iteration 300 is 0.005259
the cost at iteration 100 is 0.017934
the cost at iteration 200 is 0.0091097
the cost at iteration 300 is 0.0052618
the cost at iteration 100 is 0.014149
the cost at iteration 200 is 0.0080923
the cost at iteration 300 is 0.0050893
the cost at iteration 100 is 0.015759
the cost at iteration 200 is 0.0072318
the cost at iteration 300 is 0.0049754
the cost at iteration 100 is 0.016073
the cost at iteration 200 is 0.0074095
the cost at iteration 300 is 0.0048636
the cost at iteration 100 is 0.014149
the cost at iteration 200 is 0.0063053
the cost at iteration 300 is 0.0047262
the cost at iteration 100 is 0.011294
the cost at iteration 200 is 0.0074256
the cost at iteration 300 is 0.0045935
```

```
%% --- Validation and Plotting ---

% 1. Import Validation Data
    validation_data = readmatrix('classification_validation.csv');
    validation_data(1,:) = []; % Remove NaN

val_x1 = validation_data(:, 2);
val_x2 = validation_data(:, 3);
val_y  = validation_data(:, 4);

% 2. Generate Grid
resolution = 0.1; % adjust for smooth contours
x_min = min([feature1(:,2); val_x1]) - 0.2;
x_max = max([feature1(:,2); val_x1]) + 0.2;
y_min = min([feature2(:,2); val_x2]) - 0.2;
y_max = max([feature2(:,2); val_x2]) + 0.2;

[xx, yy] = meshgrid(x_min:resolution:x_max, y_min:resolution:y_max);
```

```
grid_points = [xx(:), yy(:)];

% 3. Predict on Grid
Z = zeros(size(grid_points, 1), 1);

for k = 1:size(grid_points, 1)
    sample_grid = grid_points(k, :)';

    [~, a_out] = forward(sample_grid, w, b);

    Z(k) = a_out{end};
end

% Reshape Z back to the grid shape for contour plotting
Z = reshape(Z, size(xx));

% 4. Create the Contour Plot
figure('Name', 'Decision Boundary', 'Color', 'w');
hold on;

% Draw the probability map
contourf(xx, yy, Z, 50, 'LineColor', 'none');
colormap(jet);
colorbar;
clim([0 1]);

% Draw decision boundary (Z = 0.5)
[C, h] = contour(xx, yy, Z, [0.5 0.5], 'k', 'LineWidth', 3);
clabel(C, h, 'FontSize', 12, 'Color', 'white');

%  Plot the Validation Data
% Plot Class 0 points
idx0 = (val_y == 0);
scatter(val_x1(idx0), val_x2(idx0), 60, 'o', 'filled', ...
    'MarkerFaceColor', 'cyan', 'MarkerEdgeColor', 'k', 'DisplayName', 'Class 0');

% Plot Class 1 points
idx1 = (val_y == 1);
scatter(val_x1(idx1), val_x2(idx1), 60, 's', 'filled', ...
    'MarkerFaceColor', 'magenta', 'MarkerEdgeColor', 'k', 'DisplayName', 'Class 1');

% Formatting
title('Validation Data & Decision Boundary');
xlabel('Feature 1');
ylabel('Feature 2');
legend('Location', 'best');
grid on;
hold off;
```




```
%% --- Hyperparameter Sweep ---
%constants -> change as necessary
if runhyperparamsweep == 1

neurontrain = 10:10:250;
inputfeatures = 2;
layernumber = 1; %code can be generalized to arbitrary layer number
rng(12345)
neuronindex = 1;
alphatrain = 0.01:0.01:0.25; %learning rate;
epsilon = 1e-15; %avoid division by 0;
epochtrain = 25;
val_X = [val_x1 val_x2];
    total_val = length(val_X);
for neuronperlayer = neurontrain

alphaindex = 1;
for alpha = alphatrain

%creating weights of 1st layer
wtrain{1} = 0.5 * rand(neuronperlayer,inputfeatures) - 0.25;

%initialise Weights and biases of additional layers
```

```
for i=1:layernumber
    btrain{i} = zeros(neuronperlayer,1);
end

if layernumber > 1

    for i = 2:layernumber
    wtrain{i} = 0.5 * rand(neuronperlayer,neuronperlayer) - 0.25;
    end
end

%initialise Weights and biases of final layer

wtrain{layernumber + 1} = 0.5 * rand(outputneurons,neuronperlayer) - 0.25;
btrain{layernumber + 1} = zeros(outputneurons, 1) ;




for i=1:epochtrain
%shuffle data so that every epoch has a different dataset
    shuffler = randperm(length(trainingdata));
    clear cost cumulcost

for j=1:length(trainingdata)
%feedforward through dataset one sample at a time
samplex = [feature1(shuffler(j),2); feature2(shuffler(j),2)];
sampley = [trainingclass(shuffler(j),2)];

[n, a] = forward(samplex, wtrain, btrain);

[wtrain, btrain] = backward(samplex, sampley, n, a, wtrain, btrain, alpha,
layernumber) ;

%evaluate cost
cost(j) = -(sampley * log(a{end} + epsilon) + (1 - sampley) * log(1 - a{end} +
epsilon));
%cost check every iteration

if mod(j,100) == 0
    cumulcost = sum(cost)/j;
    %disp(['the cost at iteration ' num2str(j) ' is ' num2str(cumulcost)])
end

end
end
finalcost(alphaindex,neuronindex) = cumulcost;

%check accuracy against validation data
```

```
% put a forward pass here on the validation data
    correct_count = 0;

    for v = 1:total_val
                vx = val_X(v,:)';
                vy = val_y(v);

                [~, va_out] = forward(vx, wtrain, btrain);
                prediction = va_out{end} >= 0.5;

                if prediction == vy
                    correct_count = correct_count + 1;
                end
    end

            accuracy(alphaindex, neuronindex) = (correct_count / total_val) * 100;

            alphaindex=alphaindex+1;

end
neuronindex = neuronindex + 1;
end

mincost = min(finalcost, [], "all");
[mincostrow, mincostcol] = find(finalcost == mincost);

maxacc = max(accuracy, [], "all");
[maxaccrow, maxacccol] = find(accuracy == maxacc);

alphacost = alphatrain(mincostrow);
neuroncost = neurontrain(mincostcol);

alphaacc = alphatrain(maxaccrow);
neuronacc = neurontrain(maxacccol);

disp(['The minimal cost learning rate is ' num2str(min(alphacost)) ' and the neuron
number is ' ...
    num2str(min(neuroncost))])

disp(['The maximum accuracy learning rate is ' num2str(min(alphaacc)) ' and the
neuron number is ' ...
    num2str(min(neuronacc))])

%at the end of every alpha loop, check the training data

end
```

```
The minimal cost learning rate is 0.25 and the neuron number is 10
The maximum accuracy learning rate is 0.06 and the neuron number is 10
```

```
%% Backwards Func

function [w, b] = backward(x, y, n, a, w, b, alpha, layernumber)

% 1. dLoss/doutput -> BCE with sigmoid means it's guess - true
dl_dout{layernumber + 1} = a{end} - y;

%2. gradients of activation functions
%gradient of tanh, use n1 because tanh


%gradient of sigmoid, set to 1 because our derivative of loss / output
%includes the sigmoid
fprime{layernumber + 1} = 1; %a{end}*(1-a{end}); %only for last layer

for j = 1:layernumber

fprime{j} = tanhgrad(n{j}); %same for all hidden layers

end


%3. Derivative of loss w.r.t hidden layers
for j = layernumber+1:-1:1



if j-1 > 0
dl_dout{j-1} = w{j}' * (dl_dout{j} .* fprime{j});
end


%5. calculate dl/db (or what's inside the brackets)
dl_db{j} = dl_dout{j}.*fprime{j}; %using hadamar

%5. calculate dl_dw
if j == 1
    dl_dw{j} = dl_db{j}*x';

else
    dl_dw{j} = dl_db{j}*a{j-1}';
end
end

for j=1:layernumber+1
w{j} = w{j} - alpha*dl_dw{j};
b{j} = b{j}- alpha*dl_db{j};

end
```

```
end

%% Forward Function

function [n, a] = forward(x, w, b)

layernumber = length(w);

for l = 1:layernumber

    if l == 1
        n{l} = w{l} * x + b{l};
        a{l} = tanh(n{l});


    elseif l == layernumber
        n{l} = w{l} * a{l-1} + b{l};
        a{l} = mysig(n{l});

    else
        n{l} = w{l} * a{l-1} + b{l};
        a{l} = tanh(n{l});
    end

end
end




%% Math Funcs

function grad = tanhgrad(x)

grad = 1-tanh(x).^2;

end

%sigmoid trick
function sigout = mysig(z)
    if( z>=0 )
        sigout = 1 / (1 + exp(-z));
    else
         sigout = exp(z) / (1 + exp(z));
    end
end
```