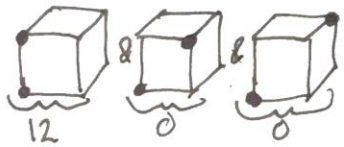
2.1) $2^{2^n} \Rightarrow 2^{2^3} = 256$

2.2)  $\Rightarrow 3$

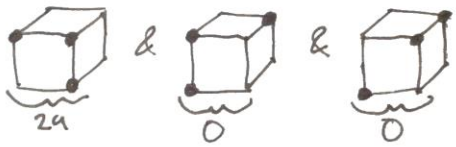
2.3) $k=0 \Rightarrow 1$ $k=8 \Rightarrow 1$ sum = 2

$k=1 \Rightarrow 8$ $k=7 \Rightarrow 8$ sum = 2+16 = 18

$k=2 \Rightarrow$ 

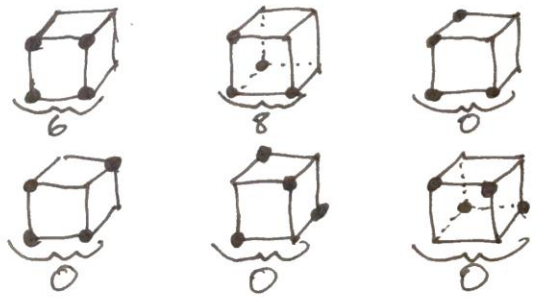
sum = 18 + 12 = 30

$k=6$ same as $k=2 \Rightarrow$ sum = 30 + 12 = 42

$k=3 \Rightarrow$ 

sum = 42 + 24 = 66

$k=5$ same as $k=3 \Rightarrow 24$ sum = 66 + 24 = 90

$k=4$ 

$\Rightarrow$ sum = 90 + 6 + 8 = 104

```
TargetA = [-1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, -1, 1, -1, -1];
TargetB = [1, -1, 1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 1, -1, -1, 1];
TargetC = [1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, -1, 1, 1, -1, 1];
TargetD = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1];
TargetE = [1, 1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1, 1, -1, -1, -1];
TargetF = [1, -1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1];


target = TargetF; %I just change this
inputs = readmatrix('input_data_numeric.csv');

%initiate random weights and threshold
xmin = -0.2;
xmax = 0.2;
numOfInputs=4;
w = xmin+rand(1,numOfInputs)*(xmax-xmin);
threshold = -1 + rand(1,1)*(2);
%----
n = 0.02;
outputs = zeros(16,1);

for  i = 1:100000
    u = randi(16);
    in = inputs(u,2:5);

    %Feed in
    in = in';
    Bi = (w*in)-threshold;
    outputs(u) = tanh((1/2)*Bi);

    %Update
    derBi = ( (sech(Bi/2))^2 )* (1/2);
    weightedError = (target(u) - outputs(u))*derBi;

    inTmp = in';
    deltaW = n*weightedError*inTmp;
    w = w + deltaW;


    %Check outputs
    if(mod(i,16) == 0)
        tmpOutputs = outputs';
        for u2 = 1:16
            if tmpOutputs(u2) >= 0
                tmpOutputs(u2) = 1;
            else
                tmpOutputs(u2) = -1;
            end
        end

        if target == tmpOutputs
            disp("SUCCESS!");
```

```
            break;
        end
    end
end
```

Published with MATLAB® R2020a

```
%------ THIS IS MAIN, BELOW IS ALL FUNCTONS. IN REALITY THEY ARE
 SEPARATE M-FILES!!

trainingSet = csvread("training_set.csv");
targetOuts = trainingSet(:,3);

v1_size = 8;
v2_size = 6;

n = 0.02;

outThreshold = 0;
v1_thresholds = zeros(v1_size,1);
v2_thresholds = zeros(v2_size,1);


wjk = randn(v1_size,2); %first
wij = randn(v2_size,v1_size); %second
wOut = randn(1, v2_size); %out

allXs = trainingSet(:,1:2);
xs;
for i = 1:1000000 %10000
     randIndx = randi([1 10000],1,1);

    %----- FORWARD PROP-------------------------
    xs = allXs(randIndx,:);
    xs = xs';

    local_v1 = (wjk*xs) - v1_thresholds;
    v1 = tanh(local_v1);

    local_v2 = (wij*v1) - v2_thresholds;
    v2 = tanh(local_v2);

    local_out = (wOut*v2) - outThreshold;
    out = tanh(local_out);

    %----- ERROR BACKPROP-----


    %error acc with outputlayer
    outError = OutputError(targetOuts(randIndx),out,local_out);
    outError;

    %error acc with v2
    v2Error = ((wOut') * outError) .* (1-((tanh(local_v2).^2)));
    v2Error;

    %error acc with v1
    v1Error = ((wij') * v2Error) .* (1-((tanh(local_v1).^2)));
    v1Error;
```

```
    %-----UPDATE WEIGHTS---------
    wjk = wjk + (n* v1Error * (xs'));
    wij = wij + (n * v2Error * (v1'));
    wOut = wOut + (n*outError * (v2'));


    %-----UPDATE THRESHOLDS----
    v1_thresholds = v1_thresholds + -(n*v1Error);
    v2_thresholds = v2_thresholds + -(n*v2Error);
    outThreshold = outThreshold + -(n*outError);
end

%----Compute C -----

validationSet = csvread("validation_set.csv");
allValXs = validationSet(:,1:2);
validationTargets = validationSet(:,3);
valIters = size(validationTargets,1);
realOs = zeros(valIters,1);

for i = 1:valIters
    valXs = allValXs(i,:);
    valXs = valXs';

    v1B = LocalFieldB(v1_thresholds, wjk, valXs, v1_size);
    v1 = tanh(v1B);

    v2B = LocalFieldB(v2_thresholds, wij, v1, v2_size);
    v2 = tanh(v2B);

    outB = OutputB(outThreshold, wOut, v2);
    out = tanh(outB);

    realOs(i) = out;
end

C = ClassificationError(realOs,validationTargets,valIters);
C
csvwrite('w1.csv',wjk);
csvwrite('w2.csv',wij);
csvwrite('w3.csv',wOut);
csvwrite('t1.csv',v1_thresholds);
csvwrite('t2.csv',v2_thresholds);
csvwrite('t3.csv',outThreshold);



%----------- END OF MAIN, FOLLOWING ARE FUNCTIONS------------


function c = ClassificationError(os, ts,u)
errors = 0;
```

```
% os
signedOs = zeros(u,1);
    for i = 1:u
        o = os(i);
        if o >= 0
            o = 1;
        else
            o = -1;
        end
        signedOs(i) = o;
    end
%     signedOs
%     ts
    for i = 1:u
        o = signedOs(i);
        t = ts(i);
        if(o ~= t)

            errors = errors + 1;
        end
    end
    c = errors/u;
end

% ------------------------------

function v = LocalFieldB(thresholds,weights,xs,neuronSize)

    v = zeros(neuronSize,1);

    for j = 1:neuronSize
        rowWeights = weights(j,:);
        thresJ = thresholds(j);
        vj = -thresJ + (rowWeights*xs);
        v(j) = vj;
    end
end

%-----------------

function o = OutputB(threshold, weights, v2)
    sum = weights*v2;
    sum = -threshold + sum;
    o = sum;
end

% -------------------

function delta = OutputError(t,o,b)
    delta =(t-o)*(1-(tanh(b))^2);
end

%---------
```

Published with MATLAB® R2020a