Lukas Fu Homework 1

Boolean Functions

Lukas Fu

The following table shows how many unique boolean functions the program was able to compute, as well as how many of them were linearly separable, for each value of n. Since the number of linearly separable functions change every run, the result of an arbitrarily chosen run is presented.

n	number of boolean functions	linearly separable
2	16	9
3	256	128
4	9335	4760
5	10000	5187

Tabell 1: For each number n, the number of unique boolean functions and the number of linearly separable boolean functions are listed. The maximum number of unique boolean functions possible is limited to 10000 since that is the number of trials being run.

Comparing to some numbers found online regarding the number of unique boolean functions and how many are linearly separable, my results for $n = 3, 4$ are higher while $n = 2, 5$ are lower. Currently my code, once an input has been tested, it can not be tested again. It also seems possible that due to some randomness in the weights, some functions are incorrectly classified. Therefore any classification, whether correct or not, stays in the result.

Something I tested afterwards was changing the input/output booleans to 1/0 instead of +1/-1 to see it would change things. Overall the number of linearly separable functions are significantly reduced, mayhaps as it is supposed to be. The following is a table is equivalent to table 2, except that the input and output are 1/0.

n	number of boolean functions	linearly separable
2	16	4
3	256	45
4	9270	1186
5	10000	947

Tabell 2: The same table as table 2 except that the Matlab code has been changed so that input and output booleans are 1/0 instead of +1/-1

To me this result is more reasonable where the quantities are undershot rather than overshot, since it is more reasonable to expect randomness to cause one error rather than cause 20 lack of errors. Even for $n = 5$ where we expect a huge number of linearly separable functions, the sheer quantity of boolean functions at that value of n makes it far more unlikely for a found function to linearly separable for our limited number of trials.

However in this case $n = 2, 3$ show a success rate of finding linear separability to be a bit low, under 50%. Still this is better than finding more linearly separable functions than there are supposed to be, when inaccuracies should cause errors that stop separable functions from being counted, rather than allow inseparable functions to be counted.

Boolean Functions - Matlab Code

Lukas Fu

Matlab Code

```
clear
clc
n=3; % #dimensions 2,3,4,5
nTrials = 10000; nEpochs = 20;
eta = 0.05; % learning rate
counter = 0;
%booleanInputs = [0 0 ; 0 1 ; 1 0 ; 1 1];
booleanInputs = [];
for D = 1:2^n
    binNum = dec2bin(D-1,n);
    binNumArray = regexp(binNum,'\d','match');
    binVec = [];
    for i = 1:n
        binDigit = str2double(binNumArray{i});
        binVec = [binVec binDigit];
    end
    booleanInputs = [booleanInputs ; binVec];
end
% for i = 1:numel(booleanInputs) % set 0 to -1
%     if booleanInputs(i) == 0
%         booleanInputs(i) = -1;
%     end
% end
usedBool = {};
duplicateCounter = 0;
for trial = 1:nTrials
    %sample boolean function
    booleanOutput = randi([0 1],2^n,1); % random vector of 1 and 0
%     for i = 1:length(booleanOutput) % set 0 to -1
%         if booleanOutput(i) == 0
%             booleanOutput(i) = -1;
%         end
%     end
    isNotMember = true;
    for l = 1:length(usedBool)
        if usedBool{l} == booleanOutput
            isNotMember = false;
            duplicateCounter = duplicateCounter +1 ;
            break
        end
    end
    if isNotMember %if output not in usedBool
        w = randn(1,n) / sqrt(n); %weight
```

```
44          th = 0; %threshold
45
46          for epoch = 1:nEpochs
47              for mu = 1:2^n % compute output
48                  totalError = 0;
49                  b = 0;
50                  for j = 1:n
51                      b = b + w(j) * booleanInputs(mu,j);
52                  end
53                  y = sign(b-th);
54                  error = booleanOutput(mu) - y;
55                  %update weight and threshold
56                  dw = eta * (error) * booleanInputs(mu,:);
57                  dth = -eta * (error);
58                  w = w - dw;
59                  th = th - dth;
60                  totalError = totalError + abs(error);
61              end
62              if totalError == 0
63                  counter = counter+1;
64                  break
65              end
66          end
67      end
68      usedBool{end+1} = booleanOutput; % add booleanOutput array to usedBool
            array
69  end
70  numberOfBoolFunc = nTrials - duplicateCounter;
71  disp(numberOfBoolFunc)
72  disp(counter)
```