Boolean function

```
clear all;
clc;
% Variable Initialization

nDimensions = 5;
eta = 0.05;
nTrials = 10^4;
nEpochs = 20;
bool_output_used = [];
func_count = 0;

% Input function Initialization
if nDimensions == 2
    boolean_input_function = [0 0 1 1; 0 1 0 1];
elseif nDimensions == 3
    boolean_input_function = [0 0 0 0 1 1 1 1; 0 0 1 1 0 0
1 1; 0 1 0 1 0 1 0 1];
elseif nDimensions == 4
    boolean_input_function = [0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
1; 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1; 0 0 1 1 0 0 1 1 0 0 1 1
0 0 1 1; 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1];
elseif nDimensions == 5
    boolean_input_function = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1; 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1; 0 0 0 0 1 1 1 1 0
0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1; 0 0 1 1 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1; 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1];
end


% Generating output boolean patterns
for nTrials = 1: nTrials

    boolean_output = 2 * randi([0, 1], 1, 2^nDimensions)-1;


% Checking if the boolean ouptput is used or not
    c = 0;
    for i = 1: size(bool_output_used,1)
```

```
        if isequal(boolean_output, bool_output_used(i,:))
            c = c+1;
        end
    end

    if c == 0
% Training the perceptron

        weight_vec =
randn(nDimensions,1)*(1/sqrt(nDimensions));
        theta = 0;
        for j = 1:nEpochs
            total_error = 0;
            for k = 1 : 2^nDimensions
                weight = 0;
                for p =1 : nDimensions
                    weight = weight +
weight_vec(p,1)*boolean_input_function(p,k);
                end
                local_field = (weight - theta);
                if local_field == 0
                    local_field = 1;
                end
                updated_output = sign(local_field);
                err = (boolean_output(:,k) -
updated_output);
                delta_w =
eta*(err)*boolean_input_function(:,k);
                delta_theta = -eta*(err);
                weight_vec = weight_vec + delta_w;
                theta = theta + delta_theta;
                total_error = total_error + abs(err);
            end
            if total_error == 0
                func_count = func_count +1;
            break;
            end
        end
        bool_output_used = [bool_output_used;
boolean_output];
    end
end
disp(func_count)
```

Dimension	No of Boolean functions	Linearly Separable functions	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
2	16	14	14	14	14	14	14
3	256	104	102	103	104	104	104
4	65536	1882	245	255	267	238	224
5	4294967296	94572	0	0	0	0	0

Table: Number of linearly separable functions in different dimensions (Source- Wikipedia)

I have mentioned the outputs i.e., the number of linearly separable functions identified in 5 different trials. For dimension 2 and 3 the output is satisfactory, however if I increase the number of epochs for dimension 3, I get the output as 104 in the first trial itself. Also, the number of trials i.e., 10,000 are good for the dimension 2 and 3 but when we go to the dimension 4 and 5, it becomes difficult because the number of Boolean functions possible itself is greater than the trials. So, it is not possible to identify all the functions. This can also be verified with the ratio of the linearly separable functions to the number of Boolean functions. Therefore, the output generated will always be less than the actual number possible with these parameters.