1 Discussion of result

From the code I have written, the number of linearly separable boolean functions for dimensions n = 2, 3, 4, 5 is shown in the following table:

Number of dimensions n	Number of linearly separable boolean functions
2	14
3	104
4	206
5	0

Table 1: Number of linearly separable boolean functions for different number of dimensions

First I will explain the code briefly(will not go in detail), and from that I will discuss the result that is given in table 1. The code creates boolean inputs, that is all possible combination of 1 and -1 for n-dimensions. For instance if the dimension is n=2 then the boolean inputs is a vector containing all possible combination of 2-value pairs of 1 and -1 which is $2^n = 4$ possible combinations. For n=3 we have a input vector containing all 3-values pair of 1 and -1 which is $2^3 = 8$ possible combinations. after that 10^4 trials is made. For every trial random boolean outputs with 1 and -1 is generated. Than a loop for 20 epochs is made where for every epoch we are calculating the output using equation (5.9) from the course book. If the output differs from the target then we have an error and we have to update the weights and threshold according to the learning rules stated in the task description at OpenTa. When the output for a tested boolean function meet all the targets then we have a linearly separable boolean function and we count it in a counter. When a function is tested it is stored in a list. The list is then used to prevent testing the same function. The result for n = 2 and n = 3 matches exaclty, it's exactly correct. Comparing with n = 4 and n = 5 we do not get valid results because we only have 10^4 trials and the number of boolean functions is 2^{2^n} so it's 65536 and 4294967296 boolean functions for n = 4 and n = 5 respectively. We also for every trial generate randomly a boolean function which means that if we want to randomly generate all functions for n = 4 and n = 5 we have to increase the number of trials tremendously. For example consider n = 4, we need at least as many trials, in other words trials = 65536. But we are generating these function randomly which means that we with very high probability will generate a already generated function. Therefore we need the number of trials to be much much bigger than the number of boolean functions, $trials >> 65536$ to be sure that we are testing for all possible boolean functions, but I don't now how much bigger we need, I just know that for n = 4 and n = 5, we will get different number of linearly separable boolean functions for many of the runs of the code because of the number of trials and that we generate the boolean function randomly for every trial.