Report of HM1 Boolean Function

Siyu Hu, gushusii@student.gu.se

1 Solution

Dimension (n)	2	3	4	5
Proportion of linearly separable (p)	0.875	0.40625	0.026187642	0

Table 1: Proportion of linearly separable Boolean functions for different dimensions

Table 1 shows the proportion of linearly separable Boolean functions for different dimensions $n = 2, 3, 4, 5$. From the table, as the dimension n increases, the proportion of linearly separable Boolean functions decreases. It indicates that with the increase in input dimension, more Boolean functions become non-linearly separable. In linear space, as the dimension n increases, the complexity of data point distribution also rises, so more Boolean functions cannot be separated. Especially in higher dimensions, the combinations and distributions of samples exhibit more complex patterns, making it harder for the perceptron to find a unique line to distinguish all input-output patterns.

2 Defect Description

2.1 Accuracy of Results

For n-dimensional input, the total number of unique input-target pairs is $2^{(2^n)}$. For $n = 3$ and $n = 4$, these 10,000 samples are essentially random selections from the pool of unique samples. As a result, the proportion of linearly separable cases may slightly fluctuate with each program execution.
When running the program multiple times, I noticed that even for $n = 1$ and $n = 2$, the proportion of linearly separable functions occasionally varies slightly. After debugging, I found that the `count_linearly_separable` variable sometimes misses 1 or 2 linearly separable samples. This might be due to the 10,000 samples not fully covering all $2^{(2^n)}$ unique functions. However, theoretically, the probability of missing unique samples should be very small. I'm unsure if there are other underlying issues causing this behavior. Due to time constraints, I have submitted this report along with the source code without conclusion.

2.2 Program Efficiency Concerns

In each iteration over `num_samples`, the function `GenerateSamples.m` creates the samples and checks for uniqueness using `ismember`. This approach can be inefficient, as `ismember` compares each new sample against the entire array of previously identified unique functions, which grows with each iteration.

`GenerateSamples.m` generates 10,000 samples for each dimension n. While the number of unique samples for $n = 2$ and $n = 3$ is actually less than 10^4, I still generate 10,000 samples, which could affect the program's efficiency.

3 Programming Approach

3.1 BooleanMainFunction

For each dimension n ($n = 2, 3, 4, 5$): Use the `GenerateSamples` function to create 10,000 input-target pairs. For each unique sample, train the perceptron for 20 epochs, updating weights and the threshold after each epoch.After training, check if the perceptron outputs match the target values. If so, the sample is considered linearly separable. Calculate the proportion of linearly separable samples as:

$$\text{proportion} = \frac{\text{count\_linearly\_separable}}{\text{number of unique samples}}$$

3.2 GenerateSamples

This function generates 10,000 random Boolean function samples for each n: generate all possible input combinations (i.e., 2^n inputs). Assign random outputs (-1 or 1) to each input. Repeat until 10,000 unique input-target pairs are obtained.