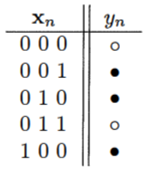
Homework Assignment – Intro to Machine Learning- CS536 - 02

Saiprakash Nalubolu | B01037579 | snalubolu@binghamton.edu

Exercise: 1.7 A Boolean target function over a three dimensional input space $\chi = \{0,1\}^3$ is given.

A Dataset D is given as follows, x_n has 3 possible input columns and 5 sample inputs are given. Output is a binary representation by white dot and a black dot for visual clarity.

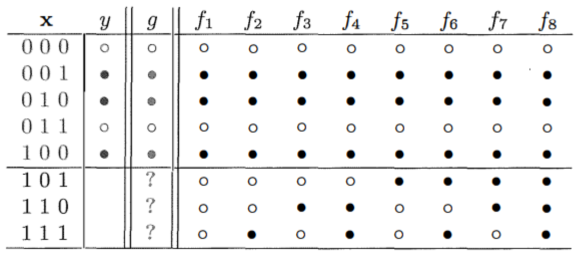


$\mathbf{x}_n$	y_n
0 0 0	○
0 0 1	●
0 1 0	●
0 1 1	○
1 0 0	●

$y_n = f(\,x_n)$ for n=1,2,3,4,5.

There are a total of 8(2^3) possible combinations for space χ since it is a three dimensional input space.

A table has been made with 8 functions(f1, f2, f3, f4, f5,f6,f7,f8) which we can use to predict the remaining 3 output values with the help of learning and a possible g(obtained from dataset D)

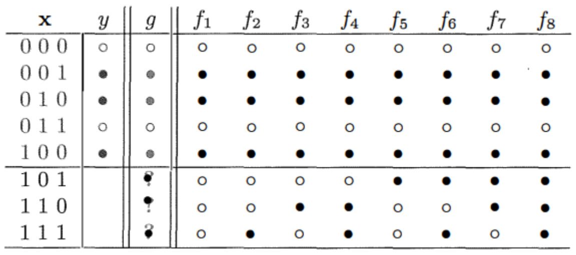


$\mathbf{x}$	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	○	○	○	○	○	○	○	○	○	○
0 0 1	●	●	●	●	●	●	●	●	●	●
0 1 0	●	●	●	●	●	●	●	●	●	●
0 1 1	○	○	○	○	○	○	○	○	○	○
1 0 0	●	●	●	●	●	●	●	●	●	●
1 0 1		?	○	○	○	○	●	●	●	●
1 1 0		?	○	○	●	●	○	○	●	●
1 1 1		?	○	●	○	●	○	●	○	●

To measure the performance of possible target functions, How many of the 8 possible target functions agree with g on all three points, on two of them, on one of them and on none of them are to be determined.

(a) H has only two hypothesis, one that always returns 'black dot(•)'and the other only returns 'white dot(o)'. The learning algorithm picks the hypothesis that matches data set the most,

- Hence, **Hypothesis that always returns '•'**



$\mathbf{x}$	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	○	○	○	○	○	○	○	○	○	○
0 0 1	●	●	●	●	●	●	●	●	●	●
0 1 0	●	●	●	●	●	●	●	●	●	●
0 1 1	○	○	○	○	○	○	○	○	○	○
1 0 0	●	●	●	●	●	●	●	●	●	●
1 0 1		●	○	○	○	○	●	●	●	●
1 1 0		●	○	○	●	●	○	○	●	●
1 1 1		●	○	●	○	●	○	●	○	●

1. Agrees with g on all three points at 1 out of 8 possible functions f8 (because f8 has only black dots as output)
2. Agrees with g on 2 points at 3 out of 8 possible functions f4,f6,f7(because they have 2 black dots and 1 white dot as output)
3. Agrees with g on a point at 3 out of 8 possible functions f2,f3,f5(because they have 1 black dot and 2 white dots as output))
4. Does not agree with any of them at only one function which is f1(all three white dots as output).

- **Hypothesis that always returns '**o**'**
  Similar to the above g only has white dots. Therefore,

1. The hypothesis agrees with g on all three points at 1 out of 8 possible functions f1 (because f1 has only white dots as output)
2. Agrees with g on two points at 3 out of 8 possible functions f2,f3,f5(because they have 2 white dots and 1 black dot as output)
3. Agrees with g on a point at 3 out of 8 possible functions f4,f6,f7 (because they have 1 white dot and 2 black dots as output))
4. Does not agree with any of them at only one function which is f8 (all three black dots as output).

(b) Now the same H but the learning algorithm picks the hypothesis that matches the Dataset the least.

- **Hypothesis that always returns '•'**

1. Agrees with g on all three points at 1 out of 8 possible functions f1 (because f1 has no black dots as output)
2. Agrees with g on two points at 3 out of 8 possible functions f2,f3,f5(because they have 1 black dot and 2 white dots as output)
3. Agrees with g on a point at 3 out of 8 possible functions f4,f6,f7(because they have 2 black dots and 1 white dot as output))
4. Does not agree with any of them at only one function which is f8(all three black dots as output).

- **Hypothesis that always returns '**o**'**
  Similar to the above g only has white dots. Therefore,

1. The hypothesis agrees with g on all three points at 1 out of 8 possible functions f8 (because f8 has no white dots as output)
2. Agrees with g on two points at 3 out of 8 possible functions f4,f6,f7(because they have 1 white dot and 2 black dots and as output)
3. Agrees with g on a point at 3 out of 8 possible functions f2,f3,f5 (because they have 1 white dot and 2 black dots as output))
4. Does not agree with any of them at only one function which is f1 (all three white dots as output).

(c) Now H is given by {XOR}, XOR= '•', if number of 1s in x is odd and is equal to 'o' if number of 1s in x is even. The below table for data out of dataset D can be derived from the given conditions of XOR. Where the first column is x_n where n is 6,7,8. Second column is expected g(x) and the third one is the functions set f1 to f8.

x	g	f1	f2	f3	f4	f5	f6	f7	f8
101	0	○	○	○	○	●	●	●	●
110	0	○	○	●	●	○	○	●	●
111	●	○	●	○	●	○	●	○	●

1. The hypothesis agrees with g on all three points at 1 out of 8 possible functions f2 (same alignment 2 white dots and one black dot)
2. Agrees with g on two points at 3 out of 8 possible functions f1(first 2 white dots), f4(first white dot and third black dot), f6(second white dot and third black dot)
3. Agrees with g on a point at 3 out of 8 possible functions f3(first white dot), f5(second white dot),f8(third black dot).
4. Does not agree with any of them at only one function which is f7 (no dots have similarity).

(d) Now the learning algorithm picks the hypothesis that agrees with all training examples but other wise disagrees the most with the XOR. Hence g can be written as,

x	g	f1	f2	f3	f4	f5	f6	f7	f8
101	●	○	○	○	○	●	●	●	●
110	●	○	○	●	●	○	○	●	●
111	o	○	●	○	●	○	●	○	●

1. The hypothesis agrees with g on all three points at 1 out of 8 possible functions f7 (two black and one white dot, same alignment)
2. Agrees with g on two points at 3 out of 8 possible functions f3(second black dot and third white dot), f5(first black dot and third white dot), f8(first and second black dots).
3. Agrees with g on a point at 3 out of 8 possible functions f1(third white dot), f4(second black dot), f6(first black dot).
4. Does not agree with any of them at only one possible function f2(unmatched).

Exercise: 1.8

Given, a random sample is picked from a bin of red and green marbles. An unknown albeit μ is defined as probability of existence of red marbles in the bin. For example, in a sample of 10 marbles drawn 3 are red, then ν is defined as fraction of red marbles which is equal to 0.3.

Probability of a red marble getting picked from the bag μ = 0.9. Probability of ν <=0.1 is to be determined.

Which means, Out of 10 marbles 0 or 1 red marble should be picked up.

Probability of drawing 0 red marbles in 10 picked up marbles means drawing out all Green marbles.

Probability of drawing a green marble is 1 – (probability of drawing a red marble) which implies to 0.1.

Hence, P(Picking 0 red marbles in a sample of 10 marbles)=0.1*0.1*0.1*….0.1(10 times) = 0.1^{10}

Probability of picking up 1 red marble out of 10 marbles is,

Probability of picking up one red marble and all others should be green marbles.

- ⇨ $10C_1$ $*$0.9*0.1*0.1*0.1….*0.1(0.1 repeats 9 times) = 10 * 0.9 * 10^{-9} {$10C_1$ denotes in how many ways red can be picked as here order of picking up red is not important}

So the probability that a sample of 10 marbles will have ν <=0.1 is 0.1^{10} + 9* 10^{-9}

= 0.1* 10^{-9} + 9 * 10^{-9} = 9.1 * 10^{-9} = 0.19 *10^{-10}

Exercise: 1.9

Hoeffding Inequality: $P(|\nu - \mu| > \epsilon) \leq 2e^{-2\epsilon^2 N}$

Given, $\nu \leq$ 0.1 and μ = 0.9

$|\nu - \mu| \geq$ |0.1-0.9|, so epsilon(ϵ) is approximately equal to 0.8.

$2e^{-2\epsilon^2 N}$ = $2e^{-2(0.8)^2 10}$ ≈ 0.55*10^{-7}

So the probability that a sample of 10 marbles with $\nu \leq$ 0.1 and μ = 0.9 is ≤ 0.55*10^{-7}

The value we calculated in exercise 1.8, falls under the range of Hoeffding's inequality.

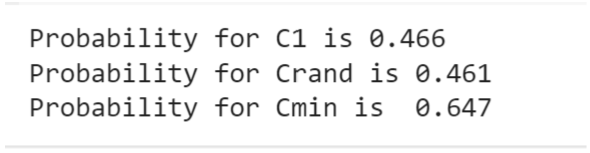
Exercise 1.10:

Computer simulation: Coinflipsimulator = https://colab.research.google.com/drive/11ibFcdNnkIybtGAoWghdtZMX6_hRaChZ

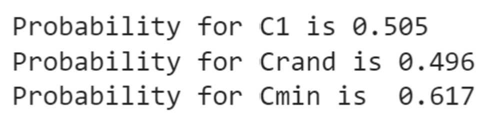
a. Probability of Heads(μ)

For C1, Number of tosses where C1 results in heads/ Total number of tosses.
For Crand, Number of tosses where Crand results in heads/ Total number of tosses.
For Cmin, Number of tosses where Cmin results in minimum number of heads/ Total number of tosses.

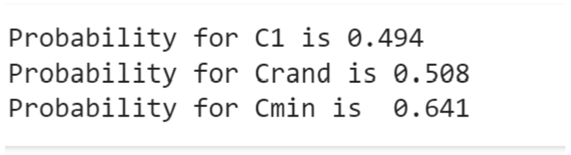
With 1000 coins and number of lips as 10, some random output sample are:

Output1: 

```
Probability for C1 is 0.466
Probability for Crand is 0.461
Probability for Cmin is  0.647
```

Output2: 

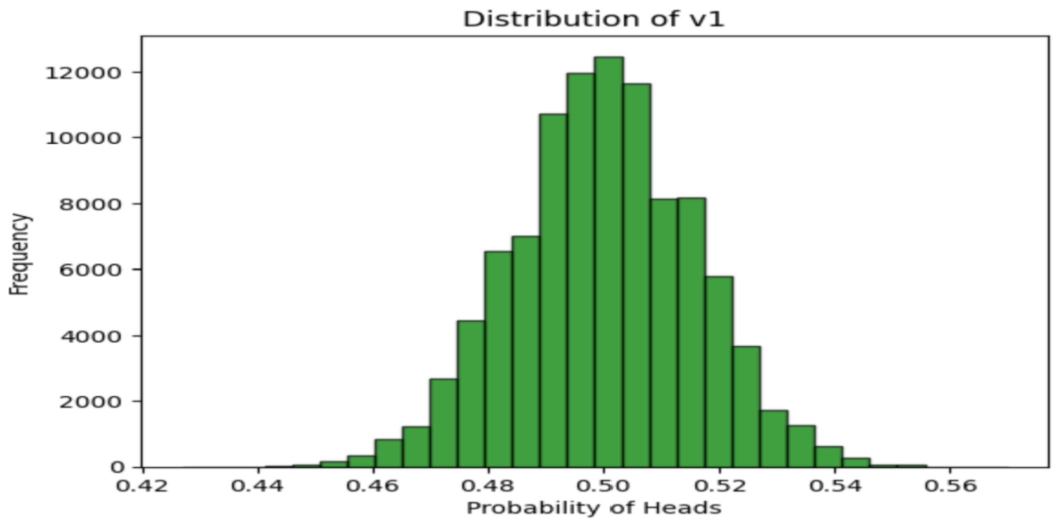
```
Probability for C1 is 0.505
Probability for Crand is 0.496
Probability for Cmin is  0.617
```

Output3: 

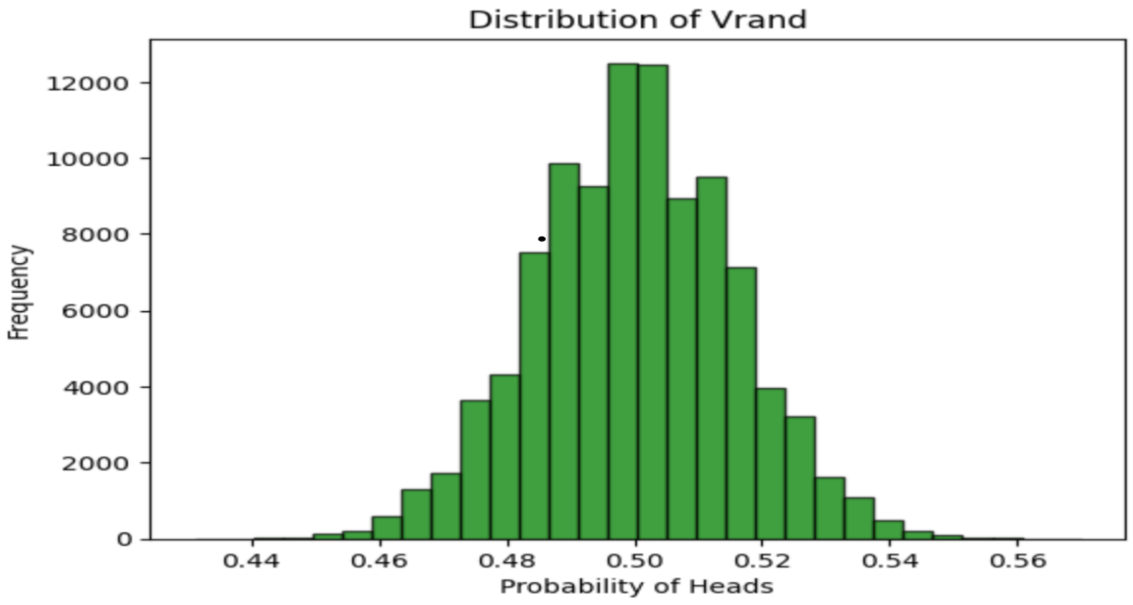
```
Probability for C1 is 0.494
Probability for Crand is 0.508
Probability for Cmin is  0.641
```

b. A sample with 10000 runs,

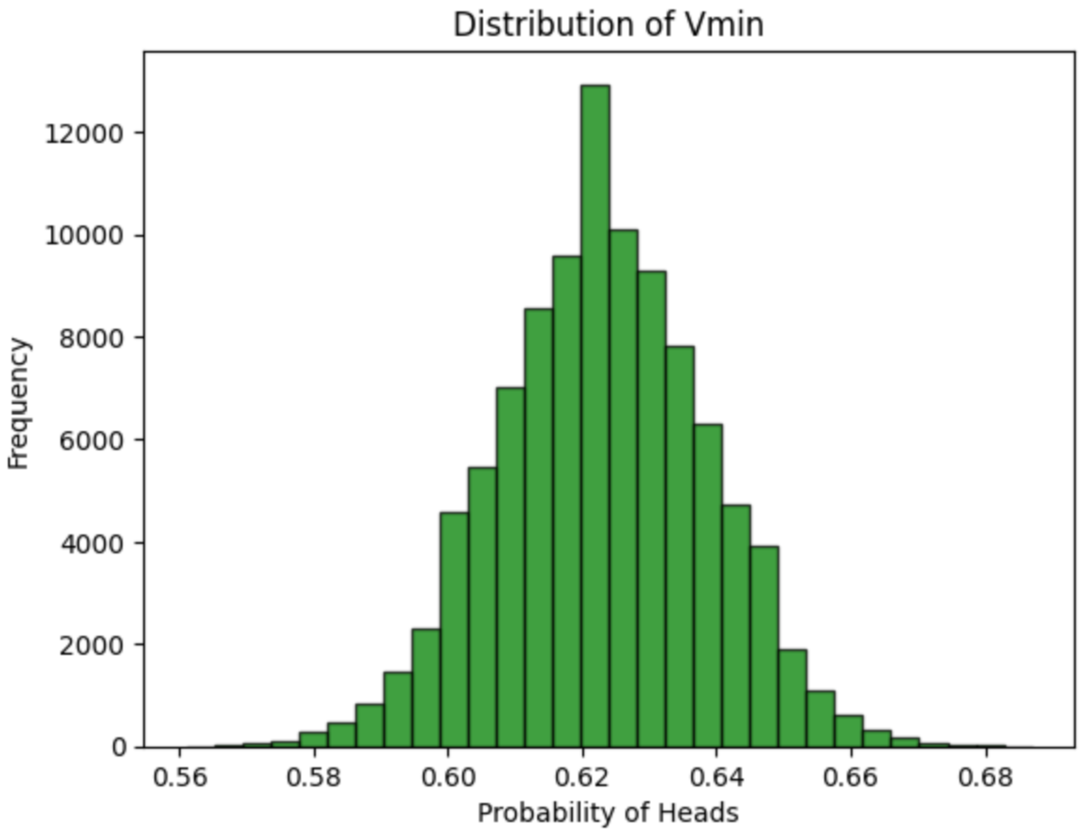
Graph for V1



Graph for Vrand



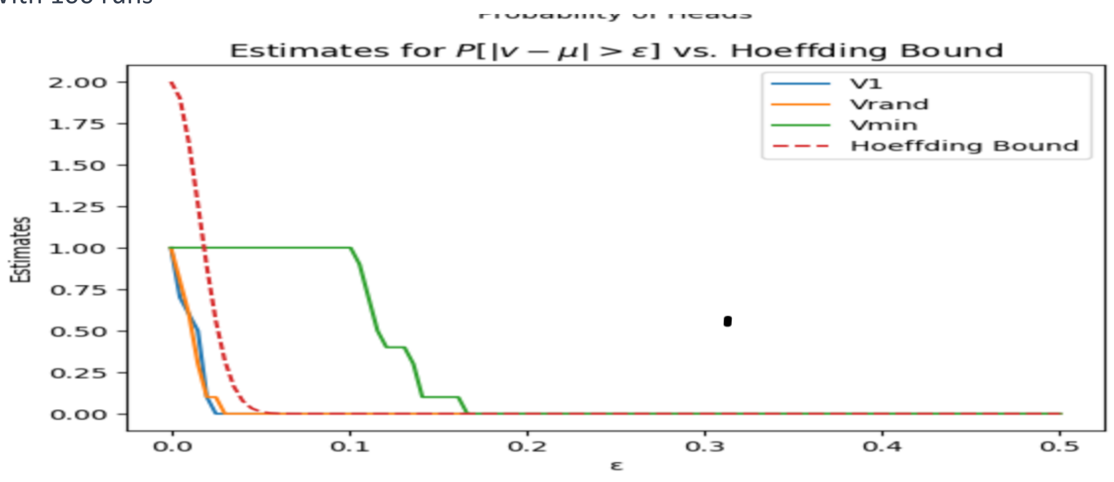
Graph for Vmin



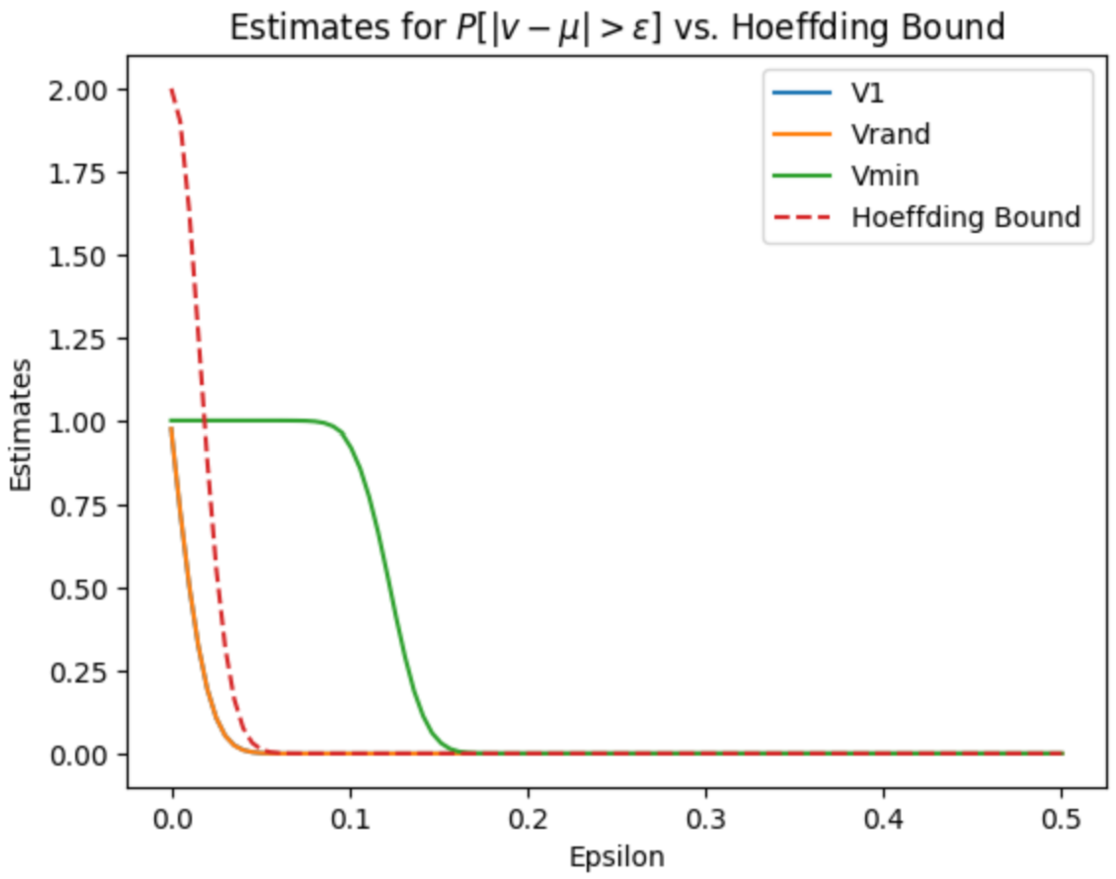
It can be inferred that the specific coins identified as Crand (coin chosen at random) and Cmin (coin with the minimum frequency of heads) can differ between runs. This variation arises from the inherent randomness involved in the selection processes. The randomness in choosing a coin at random (Crand) and the stochastic nature of identifying the coin with the minimum frequency of heads (Cmin) lead to distinct outcomes in each iteration of the experiment.

c. Plotted estimates for P[($|\nu - \mu|$)>ε] as a function of ε(threshold) Together with Hoeffding's bound

With 100 runs



With 100,000 runs



d. Coins C1 and Crand are obeying Hoeffding's bound. Cmin is not. Hoeffdings inequality is applicable to data set where μ is unknown. The selection of coin that is the coin with minimum number of heads is indicating a dependency between the trials. Here the hypothesis is getting changed while we are drawing the samples. Hence, Vmin is exceeding Hoeffding's bound in the graph. In case of V1 and Vrand hypothesis is fixed, it is not getting changed after the selection. But in case of Vmin it is getting changed and thus Vmin is exceeding the bound.

e. When we select a coin with minimum frequency of Heads, its similar to picking up a bin from M bins(Our hypothesis space) in figure 1.10. However this selection takes place after we have completed picking up the data samples, like we are picking the coin with minimum heads out of all the samples drawn. This process is comparable to a learning algorithm determining the final hypothesis. In contrast, the other two coins C1 and Crand were chosen prior to the sampling, representing a scenario where the bin is selected beforehand.

Exercise 1.12

Target function f is completely unknown. We have 4000 data points.

a. After learning, I cannot guarantee a **g** will approximate f well out of sample, As it is mentioned in the 1.3.3 Feasibility of Learning in Learning from Data Textbook "If we insist on a deterministic answer, which means that d tells us something certain about f outside D, then the answer is no". The friend is asking to guarantee g that approximates f which is impossible with out of sample data. But we can provide her with probabilistic answer. So I cannot promise her with a deterministic answer for out of sample data.

b. After learning I might not provide her a **g** along with a probability that the **g** I produce will approximate f well out of sample because the f is unknown. f might be very complex that 4000 data samples provided are not enough. And what if I am unable to get probabilistic value?

c. One of the two things will happen 1. I will produce hypothesis g (or) 2. I will declare that I failed to meet her expectations. **Well this is most likely to happen scenario and I can promise her this.** If I produce g, I will also produce a probability that approximates g to f and by Hoeffding's inequality I can promise her what she asked for. If I am unable to produce g I will declare myself that I failed to do what she had asked for. If I do return a hypothesis g, I will also give with high probability that the g which I produce will approximate f well out of sample.

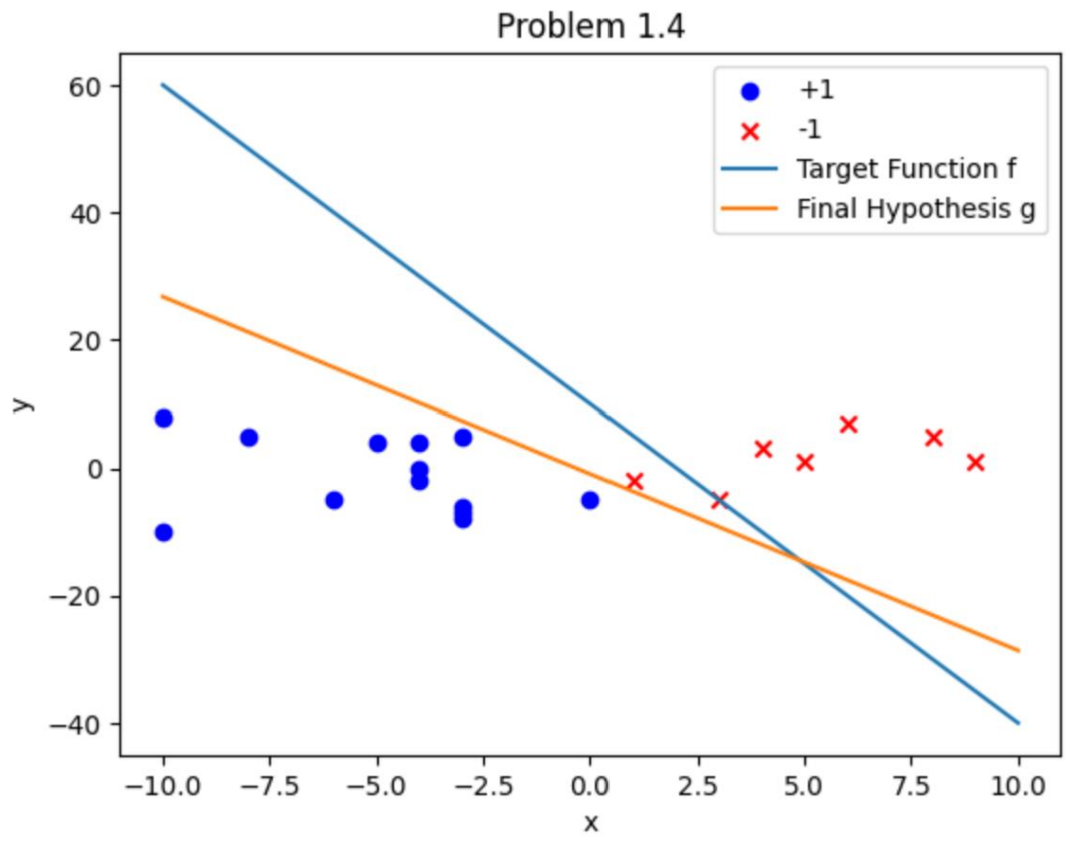
We can always evaluate in sample error Ein which is fraction of D where f and h disagrees but We cannot evaluate out of sample error Eout, it can only be estimated through probability. Probability of $g \neq f$.

Problem 1.4:

https://colab.research.google.com/drive/1E_SfvzWRCsWL1TEKxC4-BcZmeA7egcaZ#scrollTo=-RHIadG_ZgQx

a, b. Let us assume, Target function y = -5x + 10. 20 sample points are plotted and Target function, final hypothesis are plotted in same figure for a random data set xn,yn. And Number of updates needed are also provided in the output. Number of updates needed are calculated using while loop, if there are any misclassified points then the function is updated so to classify the points correctly till there are no misclassified points.

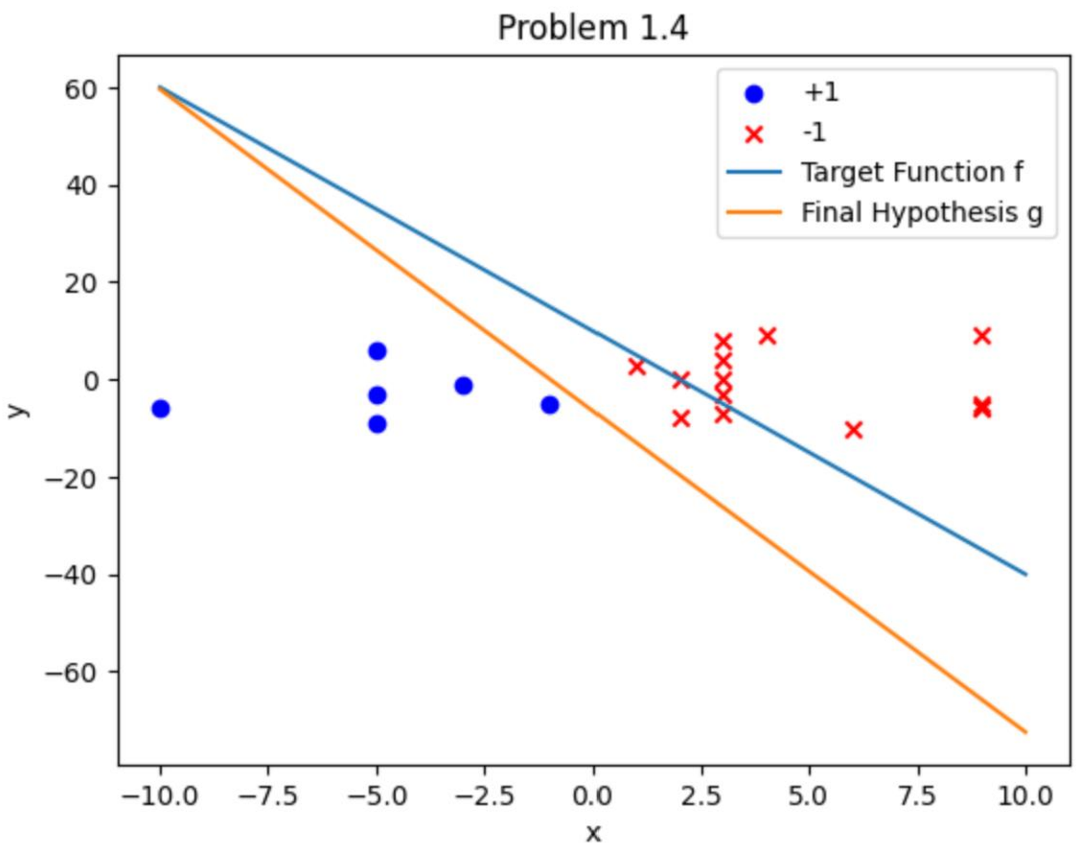
```
Perceptron learning algorithm took 3 updates before converging
```



Comment on whether f is close to g: My target function misclassified 1 red point. It took 3 updates for the algorithm to converge and make final hypothesis. The final hypothesis clearly separates all +1s and -1s. Perceptron algorithm successfully learnt a decision boundary(Orange line). It took 3 updates for the algorithm to reach final hypothesis g, which is reasonable number of updates. I feel target function was not so far In the generated random data sample.

c. Another randomly generated data of size 20,

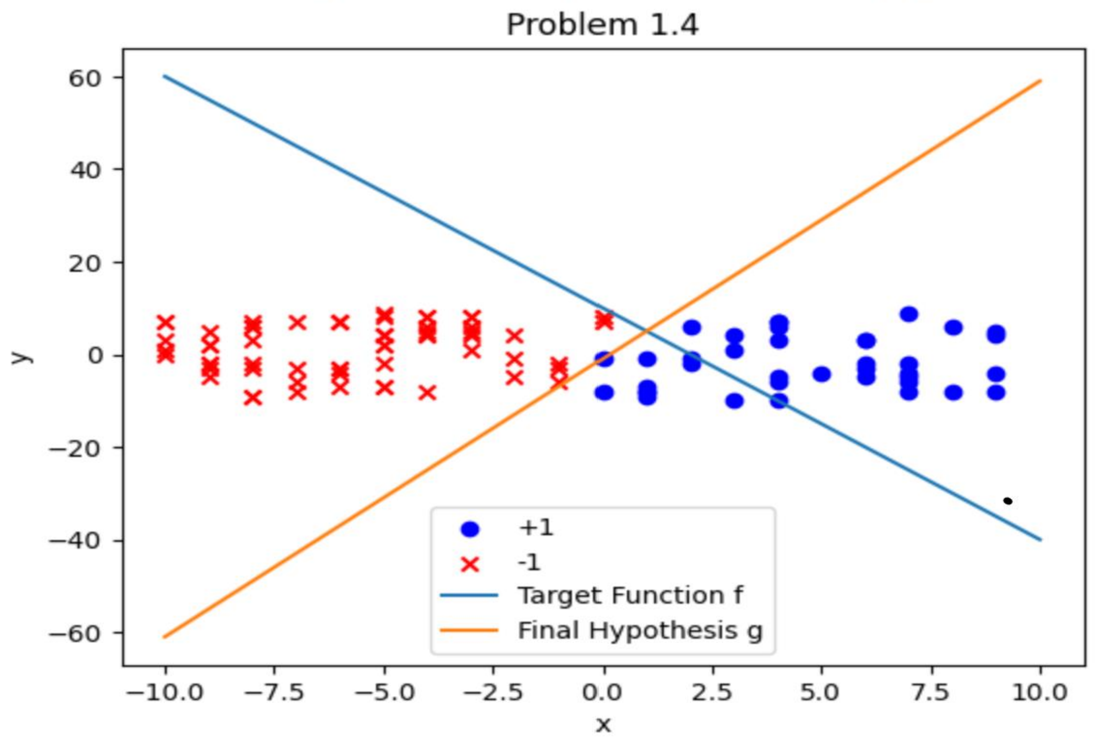
```
Perceptron learning algorithm took 4 updates before converging
```



It took 4 updates for the algorithm to achieve g in recent run(c) while it took only 3 in (b). In (c) f has 3 misclassified points while the plot in the (b) had 1. I feel though with different data in both runs f and g are close they didn't take many iteration to reach g. In both graphs Final hypothesis g is accurately segregating the points +1,-1 (or) red 'x's and blue 'o's.

d. Repeating everything in (b) with randomly generated data set of size 100.
(Plot in below page)

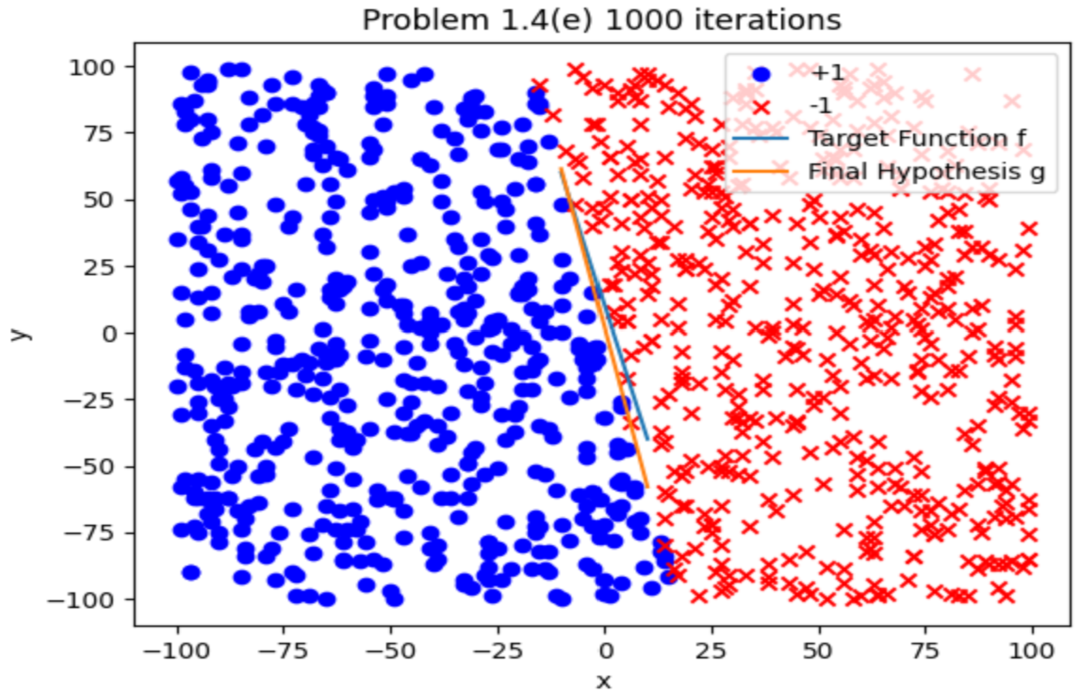
```
Perceptron learning algorithm took 19 updates before converging
```



f is very far from g as it took 19 updates.

e. Repeating everything in (b) with randomly generated data set of size 1000.

```
Perceptron learning algorithm took 19 updates before converging
```



For this 1000 data points sample it took 19 updates for convergence.

Problem 1.7:

Probability of Heads(Probability of Error) = μ

Probability of obtaining k heads in N tosses of this coin = $P[k|N,\mu] = N_{C_k}\mu^k(1-\mu)^{N-k}$

Training error ν = k/N

a. Given N = 10 and μ = 0.05, 0.8.
Probability that at least 1 coin will have ν = 0 for 1 coin, 1000 coins and 1,000,000 coins?
Now we use probability complement method to derive this.

ν = 0, implies k=0 since N≠0. Which means we have to find probability that at least 1 coin in m coins will not show any heads within N tosses.
P(ν = 0) = $10_{C_0}\mu^0(1-\mu)^{10-0}$ = $(1-\mu)^{10}$ {or $(1-\mu)^N$ $for\ N\ tosses$}

Probability that P($\nu \neq$ 0) = P($\nu >$ 0) = 1- P(ν = 0)
= 1- $(1-\mu)^{10}$

For c number of coins, P($\nu \neq$ 0) = (1- $(1-\mu)^{10}$)^c
Hence, P(ν = 0) = $1-(1-(1-\mu)^{10})^c$

So Probability of ν = 0 for N tosses and c coins is $1-(1-(1-\mu)^N)^c$

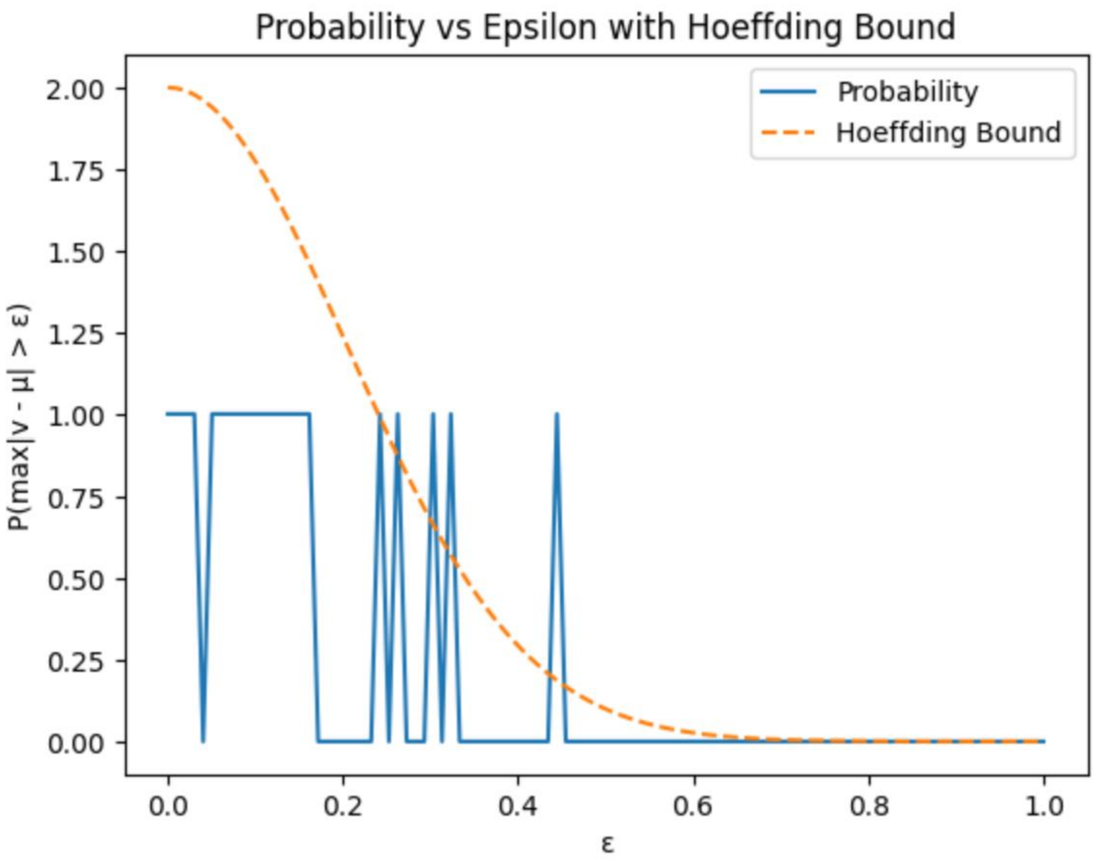
For μ = 0.05,
i. when 1 coin is tossed, m=1 => P(ν = 0) = $1-(1-(1-0.05)^{10})^1$ = 0.598
ii. when 1000 coins are tossed, m=1000 => P(ν = 0) = $1-(1-(1-0.05)^{10})^{1000}$ = 1.0
iii. when 1000 coins are tossed, m=1,000,000 => P(ν = 0) = $1-(1-(1-0.05)^{10})^{1000000}$ = 1.0

For μ = 0.8,
i. when 1 coin is tossed, m=1 => P(ν = 0) = $1-(1-(1-0.8)^{10})^1$ = 1.02400000034919e-07 $\approx$ 0.
ii. when 1000 coins are tossed, m=1000 => P(ν = 0) = $1-(1-(1-0.8)^{10})^{1000}$ = 0.00010239
iii. when 1000 coins are tossed, m=1,000,000 => P(ν = 0) = $1-(1-(1-0.8)^{10})^{1000000}$ = 0.097

For an unfair coin with probability of heads 0.8 probability of getting no heads when a single coin is tossed 10 times is very close to 0; for 1000 coins tossed 10 times is also close to 0 but getting nearer to some decimal value. And for 1000000 coins tossed 10 times is near to 0.097. As the sample space increases the output which we thought is impossible might be possible. Like there is a very very slim chance of not getting heads. But there is a chance.

Similarly above probability of heads(μ) with 0.05 can also be explained. If probability of heads is 0.05 then probability of getting tails should be 0.95. This is an unfair coin with most occurrences as tails when tossed. So probability of getting no heads is 1 when 1000 and 1,000,000 coins are tossed.

b. https://colab.research.google.com/drive/1Mko7Xm0pZ0Q9VaMQHGg0IXIoIbo1rEXy5


Probability vs Epsilon with Hoeffding Bound
Probability
Hoeffding Bound
P(max|ν - μ| > ε)
ε
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0