ASSIGNMENT 1

Deadline: 11:59 pm, February 25, 2024

Submit via Blackboard with VeriGuide receipt

Please follow the course policy and the school's academic honesty policy.
Each question is worth ten points.

1. In the weight update rule of the Perception Learning Algorithm (referring to week 1's slide page 64), we observe that the weights moves in the direction of classifying $\mathbf{x}_*$ correctly.

   The rule can be written as: $\mathbf{w}(t+1) = \mathbf{w}(t) + y_*\mathbf{x}_*$.

   Please note that we have: $\mathcal{X} = \{1\} \times \mathbb{R}^d = \left\{[x_0, x_1, \cdots, x_d]^{\mathrm{T}} \mid x_0 = 1, x_1 \in \mathbb{R}, \cdots, x_d \in \mathbb{R}\right\}$

   (1) Prove that $y_*\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}_* < 0$.

   Hint: $\mathbf{x}_*$ is misclassified by $\mathbf{w}(t)$.

   (2) Prove that $y_*\mathbf{w}^{\mathrm{T}}(t+1)\mathbf{x}_* > y_*\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}_*$.

   Hint: Use the weight update rule.

2. Consider the Hoeffding Inequality equation (referring to week 3's slide page 20) given by:

$$P[|v - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}, \text{for any } \epsilon > 0$$

   In this equation, μ denotes the probability to pick a red marble, v denotes the fraction of red marbles in sample, and N denotes the size of the sample.

   For this problem, we assume that $P[|v - \mu| > \epsilon] = \delta$, where $\delta = 0.05$. You are required to use Python's math package (i.e., import math) to calculate the results for the following questions. (Copy your python program to the answer sheet.)

   (a) Determine the minimum sample size required to ensure $\epsilon \leq 0.1$?

   (b) Determine the minimum sample size required to ensure $\epsilon \leq 0.02$?

   (c) Determine the minimum sample size required to ensure $\epsilon \leq 0.003$?

3. Consider the Hoeffding Inequality equation (referring to week 3's slide page 20) given by:

$$P[|v - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}, \text{for any } \epsilon > 0$$

   For this problem, we assume that $\mu = 0.95$. Please use the Hoeffding Inequality to bound the probability that a sample of 20 marbles will have $v \leq 0.1$. You are required to use Python's math package to calculate the results for the following questions. (Copy your python program to the answer sheet.)

4. Perception learning algorithm implementation (referring to week 4's tutorial page 16) via Python (Copy your python program to the answer sheet). Consider a perception in a two-dimensional case (d=2),

   (a) Generate a linearly separable data set of size 20. Plot the examples $(\mathbf{x}_n, y_n)$ as well as the target function f on a plane. Be sure to mark the examples from different classes differently and add labels to the axes of the plot.

   (b) Run the perceptron learning algorithm on the data set above. Report the number of updates that the algorithm takes before converging. Plot the examples $(\mathbf{x}_n, y_n)$, the target function f, and the final hypothesis g in the same figure. Comment on whether f is close to g.

   (c) Repeat everything in (b) with another randomly generated data set of size 100. Compare your results with (b).

   (d) Repeat everything in (b) with another randomly generated data set of size 1,000. Compare your results with (b).

   (e) Modify the algorithm such that it takes $\mathbf{x}_n \in \mathbb{R}^{10}$ instead of $\mathbb{R}^2$. Randomly generate a linearly separable data set of size 1,000 with $\mathbf{x}_n \in \mathbb{R}^{10}$ and feed the data set to the algorithm. How many updates does the algorithm take to converge?

   (f) Summarize your conclusions with respect to accuracy and running time as a function of N and d, where N is the size of data set and d is the data dimension.

   Hints:

   1. One way to generate a linearly separable data set: first set your own target function, then keep randomly generating data points and put each of them in the positive or negative class based on the relation between the data point and the target function.

   2. **numpy.random.randint(low, size=(N, dim)))**: return random integers from the discrete uniform distribution of the specified dtype in the half-open interval [0, low). The output shape is [N,dim].

   3. **numpy.random.choice(a, size=None, replace=True, p=None)**: generates a random sample from a given 1-D array.

   4. Some functions from Python package Matplotlib that may be useful: **matplotlib.pyplot.xlabel**, **matplotlib.pyplot.ylabel**, **matplotlib.pyplot.legend**.

   5. It is better to define some functions for perceptron experiments that can be reused in sub-questions.

5. Consider a sample of 10 marbles drawn independently from a bin that holds red and green marbles (referring to week 3's slide page 15). The probability of a red marble is μ. For $\mu = 0.05$, $\mu = 0.6$, $\mu = 0.9$, use Python to compute the probability of getting no red marbles ($\upsilon = 0$) in the following cases. (Copy your python program to the answer sheet.)

   (a) Draw only one such sample. Compute the probability that $\upsilon = 0$.

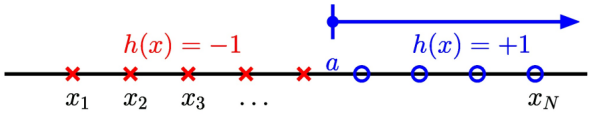
   (b) Draw 1,000 independent samples. Compute the probability that (at least) one of the samples has $\upsilon = 0$.

   (c) Repeat (b) for 1,000,000 independent samples.

   (d) Compare the results that you get from (a) to (c).

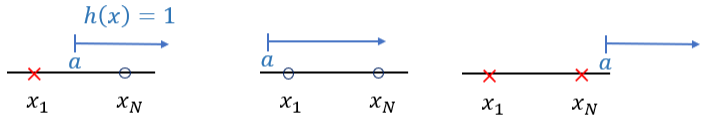
6. Compute the growth function of a hypothesis set $\mathcal{H}(N)$ (referring to week 4's slide page 25) in each of the following cases.

   (1) Positive rays: $\mathcal{H}$ consists of all hypotheses h:$\mathbb{R} \to \{-1, +1\}$ of the form $h(x) = \text{sign}(x - a)$, i.e., the hypotheses are defined in a one-dimensional input space, and they return -1 to the left of some value a and +1 to the right of a. The illustration is as follows,



Hint: When N=2, there are three different dichotomies according to its definition, see figure below:

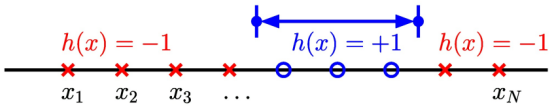
Positive rays:



$N = 2$

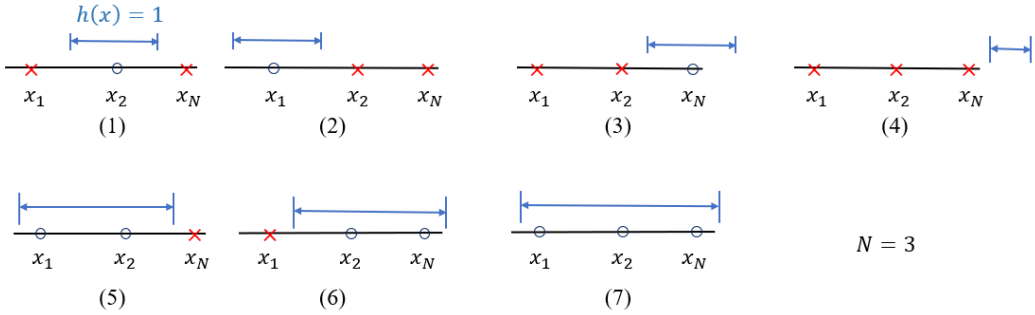
so $\mathcal{H}(2) = 3$.

(2) Positive intervals: $\mathcal{H}$ consists of all hypotheses that return +1 with some interval and -1 otherwise. Each hypothesis is specified by the two end values of that interval. The illustration is as follows,



Hint: When N=3, there are seven different dichotomies according to its definition, see the figure below:

Positive intervals:



$N = 3$

so $\mathcal{H}(3) = 7$.

*** END ***

(NOTE: my python screen shot is copied to this pdf together with my answers to problems, but some python code might be partial (eg. excluding the class definition). the full .py file is attached in the .zip file)

1. In the weight update rule of the Perception Learning Algorithm (referring to week 1's slide page 64), we observe that the weights moves in the direction of classifying $\mathbf{x}_*$ correctly.

   The rule can be written as: $\mathbf{w}(t+1) = \mathbf{w}(t) + y_*\mathbf{x}_*$.

   Please note that we have: $\mathcal{X} = \{1\} \times \mathbb{R}^d = \left\{ [x_0, x_1, \cdots, x_d]^{\mathrm{T}} \mid x_0 = 1, x_1 \in \mathbb{R}, \cdots, x_d \in \mathbb{R} \right\}$

   (1) Prove that $y_* \mathbf{w}^{\mathrm{T}}(t)\mathbf{x}_* < 0$.

   Hint: $\mathbf{x}_*$ is misclassified by $\mathbf{w}(t)$.

   proof:

   ① : y_* is positive

   since x_* is misclassified, $W^T(t)\, X_*$ has different sign with y_*, which is negative

   $\therefore y_* W^T(t)\, X_* < 0$

   ② y_* is negative

   since x_* is misclassified, $W^T(t)\, X_*$ has different sign with y_*, which is positive

   $\therefore y_* W^T(t)\, X_* < 0$

   $\Rightarrow y_* W^T(t) \cdot X_* < 0$

(2) Prove that $y_* \mathbf{w}^{\mathrm{T}}(t+1)\mathbf{x}_* > y_* \mathbf{w}^{\mathrm{T}}(t)\mathbf{x}_*$.

Hint: Use the weight update rule.

proof:

since $W(t+1) = W(t) + y_* X_*$

$\Rightarrow W^T(t+1) = W^T(t) + (y_* X_*)^T = W^T(t) + y_* X_*^T$

$\Rightarrow y_* W^T(t+1) X_* = y_* (W^T(t) + y_* X_*^T) X_*$

$= y_* W^T(t) X_* + y_*^2 X_*^T X_*$

let $X_* = (x_1, x_2, \cdots x_k)^T$

$\Rightarrow X_*^T X_* = \sum_{i=1}^{k} x_i^2 > 0$

$\Rightarrow y_*^2 X_*^T X_* > 0$

$\Rightarrow y_* W^T(t) X_* + y_*^2 X_*^T X_* > y_* W^T(t) X_*$

$\Rightarrow y_* W^T(t+1) X_* > y_* W^T(t) X_*$

2. Consider the Hoeffding Inequality equation (referring to week 3's slide page 20) given by:

$$P[|v-\mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}, \text{for any } \epsilon > 0$$

In this equation, μ denotes the probability to pick a red marble, v denotes the fraction of red marbles in sample, and N denotes the size of the sample.

For this problem, we assume that $P[|v-\mu| > \epsilon] = \overset{0.05}{\delta}$, where $\delta = 0.05$. You are required to use Python's math package (i.e., import math) to calculate the results for the following questions. (Copy your python program to the answer sheet.)

(a) Determine the minimum sample size required to ensure $\epsilon \leq 0.1$?

(b) Determine the minimum sample size required to ensure $\epsilon \leq 0.02$?

(c) Determine the minimum sample size required to ensure $\epsilon \leq 0.003$?

to achieve the minimum N, we want $\delta = 2e^{-2\epsilon^2 N}$

$\Rightarrow N = \frac{\log \frac{2}{\delta}}{2\epsilon^2}$

the python code for calculating is shown below, and the .py file is also attached with hw.

```python
import math

delta = 0.05

error = float(input('input error:'))

N_min = math.log(2 / delta) / (2 * error ** 2)

N_min = math.ceil(N_min)

print(f'N_min is {N_min}')
```

the output is

```
I3320\tutorial\hw1\q2.py" The Chinese University of Hong Kong\CUHK\y2s2\CSCI3320>
input error:0.1
N_min is 185
PS D:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSCI3320> python -u "d:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSC
I3320\tutorial\hw1\q2.py"
input error:0.02
N_min is 4612
PS D:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSCI3320> python -u "d:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSC
I3320\tutorial\hw1\q2.py"
input error:0.003
N_min is 204938
```

So the answer is,

(a): 185

(b) 4612

(c) 204938

3. Consider the Hoeffding Inequality equation (referring to week 3's slide page 20) given by:

$$P[|v - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}, \text{for any } \epsilon > 0$$

For this problem, we assume that $\mu = 0.95$. Please use the Hoeffding Inequality to bound the probability that a sample of 20 marbles will have $v \leq 0.1$. You are required to use Python's math package to calculate the results for the following questions. (Copy your python program to the answer sheet.)

Since $P(|v-\mu| > \epsilon) \leq 2e^{-2\epsilon^2 N}$

$\Rightarrow P(v-\mu > \epsilon \text{ or } v-\mu < -\epsilon) \leq 2e^{-2\epsilon^2 N}$

$\Rightarrow P(v > \mu+\epsilon \text{ or } v < \mu-\epsilon) \leq 2e^{-2\epsilon^2 N}$

let $\mu - \epsilon = 0.1$

$\Rightarrow P(v \leq 0.1) = P(v < 0.1) \leq P(v > \mu+\epsilon \text{ or } v < \mu-\epsilon) \leq 2e^{-2\epsilon^2 N}$

which means that $P(v \leq 0.1)$ has upper bound $2e^{-2\epsilon^2 N}$

the python code for calculating is shown below, and the .py file is also attached with hw.

```python
import math

mu = 0.95
N = 20
nu_max = 0.1

epsilon = mu - nu_max

upper_bound = 2 * math.exp(-2 * epsilon ** 2 * N)

print(f'upper bound is {upper_bound}')
```

the output is

```
PS D:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSCI3320> python -u "d:\OneDrive\OneDrive - The Chinese University of Hong Kong\CUHK\y2s2\CSCI3320\tutorial\hw1\q3.py"
upper bound is 5.622370597578089e-13
```

So the upper bound is 5.622×10^{-13}

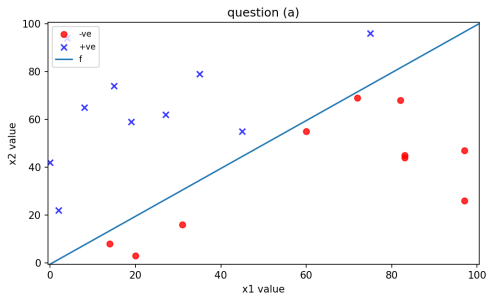
the lower bound is 0

4(a)

the python code is shown below and the .py file is also attached with hw.

```
'''
(a) generate data set of size 20
'''
size_X = 20
# set X
# X: shape(n_samples, n_features)
X = np.random.randint(0, value_max, size=(size_X, 2))

# set target function f
# f: shape (n_features + 1), f[0] is the bias term
f = np.array([0.5, -1, 1])      # 0.5 - x1 + x2

# set y
# y: shape(n_samples)
y = np.ones((20))
y = np.where(np.dot(X, f[1:])+f[0] >= 0.0,1,-1)

# plot
plot(X, y, f, 'a', 1)
```

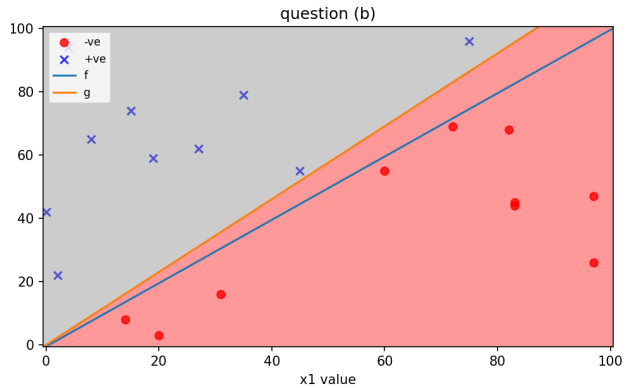
the plot is shown below



(b) the python code is shown below and the .py file is also attached with hw.

```
'''
(b) run the perceptron
'''
ppn = Perceptron(eta=0.5)
result = ppn.fit(X, y)

# number of updates
print(f"for datasize=20, number of updates is {np.sum(result.errors_)}")

# plot
plot(X, y, f, 'b', 2, result)
```

the plot and the output is shown below




```
for datasize=20, number of updates is 16
```

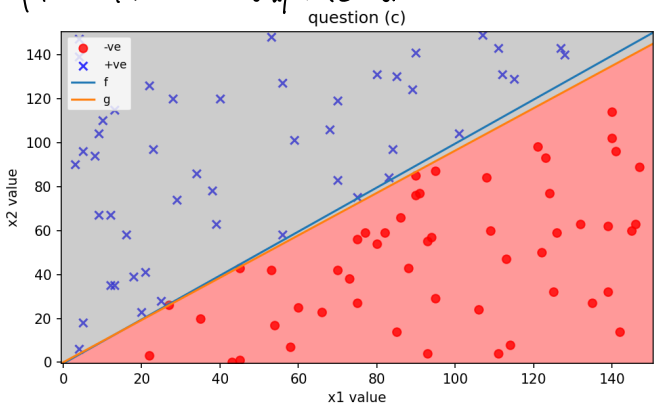
⇒ number of updates is 16.

⇒ f and g have approximately similar location and orientation, but still have a big difference between each other. f is not very close to g

(c) the python code is shown below and the .py file is also attached with hw.

```
'''
(c) run the perceptron with size=100
'''
size_X = 100
# set X
# X: shape(n_samples, n_features)
X = np.random.randint(0, value_max, size=(size_X, 2))

# set target function f
# f: shape (n_features + 1), f[0] is the bias term
f = np.array([0.5, -1, 1])      # 0.5 - x1 + x2

# set y
# y: shape(n_samples)
y = np.ones((20))
y = np.where(np.dot(X, f[1:])+f[0] >= 0.0,1,-1)

ppn = Perceptron(eta=0.5)
result = ppn.fit(X, y)

# number of updates
print(f"for datasize=100, number of updates is {np.sum(result.errors_)}")

# plot
plot(X, y, f, 'c', 3, result)
```

the plot and the output is shown below




```
for datasize=100, number of updates is 34
```

⇒ number of updates is 34, which is significantly more than that of (b)

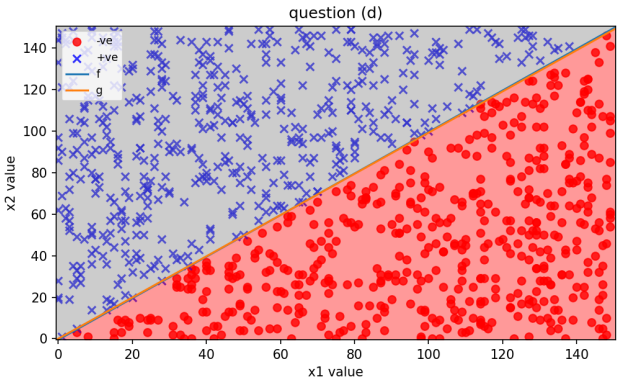
→ f and g have similar location and orientation, but on the corner of the graph there is still a bit of difference

f is a bit of close to g, which is significantly closer to f compared with g in (b)

(d) the python code is shown below and the .py file is also attached with hw.

```
'''
(d) run the perceptron with size=1000
'''
size_X = 1000
# set X
# X: shape(n_samples, n_features)
X = np.random.randint(0, value_max, size=(size_X, 2))

# set target function f
# f: shape (n_features + 1), f[0] is the bias term
f = np.array([0.5, -1, 1])      # 0.5 - x1 + x2
# set y
# y: shape(n_samples)
y = np.ones((20))
y = np.where(np.dot(X, f[1:])+f[0] >= 0.0,1,-1)

ppn = Perceptron(eta=0.5)
result = ppn.fit(X, y)

# number of updates
print(f"for datasize=1000, number of updates is {np.sum(result.errors_)}")

# plot
plot(X, y, f, 'd', 4, result)
```

the plot and the output is shown below:




```
for datasize=1000, number of updates is 175
```

⇒ number of updates is 175, which is significantly more than that of (b) and (c)

⇒ f and g have really similar location and orientation

f is very close to g (is significantly closer to f compared with g in (b) and (c))

(e) the python code is shown below, and the .py file is also attached with hw.

```
'''
(e) run the perceptron with size=1000 in 10D space
'''
size_X = 1000
# set X
# X: shape(n_samples, n_features)
X = np.random.randint(0, value_max, size=(size_X, 10))

# set target function f
# f: shape (n_features + 1), f[0] is the bias term
f = np.array([0.5, -1, 1, 2,-2, 3, -3, 4, -4, 5, -5])

# set y
# y: shape(n_samples)
y = np.ones((20))
y = np.where(np.dot(X, f[1:])+f[0] >= 0.0,1,-1)

ppn = Perceptron(eta=0.5)
result = ppn.fit(X, y)

# number of updates
print(f"for datasize=1000, dimension=10, number of updates is {np.sum(result.errors_)}")
```

the output is shown below:

```
for datasize=1000, dimension=10, number of updates is 6092
```

number of updates is 6092, which is significantly more than that of (b) and (c), (d)

(f)

from the above result we know,

when N increases, the hypothesis g will be more and more closer to f. which means that accuracy will increase

so the function could be:

$$\text{accuracy} = k_1 N \quad (k_1 > 0)$$

from the above result,

when N increases, the number of iteration increases (with d, learning rate and the range for random value fixed)

when d increases, the number of iteration increases (with N, learning rate and the range for random value fixed)

so the function could be:

$$\text{time} = k_2 \cdot N \cdot d \quad (k_2 > 0)$$

5. Consider a sample of 10 marbles drawn independently from a bin that holds red and green marbles (referring to week 3's slide page 15). The probability of a red marble is μ. For $\mu = 0.05$, $\mu = 0.6$, $\mu = 0.9$, use Python to compute the probability of getting no red marbles ($v = 0$) in the following cases. (Copy your python program to the answer sheet.)

   (a) Draw only one such sample. Compute the probability that $v = 0$.

   (b) Draw 1,000 independent samples. Compute the probability that (at least) one of the samples has $v = 0$.

   (c) Repeat (b) for 1,000,000 independent samples.

   (d) Compare the results that you get from (a) to (c).

(a)–(c): the python code is shown below and the .py file is also attached with hw.

```
sample_size = 10

'''
question (a)
'''
print("question (a)")
mu_list = [0.05, 0.6, 0.9]
pbb_list = []

for mu in mu_list:
    pbb = (1 - mu) ** sample_size
    print(f'when mu is {mu}, probability of no red marbles is :{pbb}')
    pbb_list.append(pbb)

'''
question (b)
'''
print("\nquestion (b)")
sample_num = 1000

for i, mu in enumerate(mu_list):
    pbb_all_have_red = (1 - pbb_list[i]) ** sample_num # the probability of having red marble in all samples
    pbb_have_no_red = 1 - pbb_all_have_red
    print(f'when mu is {mu}, and sample number is {sample_num}, probability of at least one has no red marbles is :{pbb_have_no_red}')

'''
question (c)
'''
print("\nquestion (c)")
sample_num = 1000000

for i, mu in enumerate(mu_list):
    pbb_all_have_red = (1 - pbb_list[i]) ** sample_num # the probability of having red marble in all samples
    pbb_have_no_red = 1 - pbb_all_have_red
    print(f'when mu is {mu}, and sample number is {sample_num}, probability of at least one has no red marbles is :{pbb_have_no_red}')
```

the output is:

```
question (a)
when mu is 0.05, probability of no red marbles is :0.5987369392383787
when mu is 0.6, probability of no red marbles is :0.00010485760000000006
when mu is 0.9, probability of no red marbles is :9.999999999999978e-11

question (b)
when mu is 0.05, and sample number is 1000, probability of at least one has no red marbles is :1.0
when mu is 0.6, and sample number is 1000, probability of at least one has no red marbles is :0.09955221269675618
when mu is 0.9, and sample number is 1000, probability of at least one has no red marbles is :1.0000000327803349e-07

question (c)
when mu is 0.05, and sample number is 1000000, probability of at least one has no red marbles is :1.0
when mu is 0.6, and sample number is 1000000, probability of at least one has no red marbles is :1.0
when mu is 0.9, and sample number is 1000000, probability of at least one has no red marbles is :9.999500844481979e-05
```

(d):

① no matter what the sample number is, the probability of having (at least one) no red marble is higher when the probability of red marble (μ) is lower

② when the number of sample increases, no matter what μ is, the probability of having (at least one) no red marble increases

③ for $\mu = 0.05$, when sample size reaches 1000, it is almost definite to have (at least one) no red marble

④ for $\mu = 0.6$, when sample size reaches 1000, the probability of having (at least one) no red marble is low (around 0.1) when sample size reaches 1,000,000 it is almost definite to have (at least one) no red marble

⑤ for $\mu = 0.9$, even when sample size reaches 1,000,000, it is still almost impossible to have (at least one) no red marble

6. Compute the growth function of a hypothesis set $\mathcal{H}(N)$ (referring to week 4's slide page 25) in each of the following cases.

(1) Positive rays: $\mathcal{H}$ consists of all hypotheses h:$\mathbb{R} \to \{-1, +1\}$ of the form $h(x) = \text{sign}(x - a)$, i.e., the hypotheses are defined in a one-dimensional input space, and they return -1 to the left of some value a and +1 to the right of a. The illustration is as follows,

N points can separate the one-dimensional input space into N+1 intervals

h can generate one dichotomy when a is inside each of the N+1 intervals.

So the number of dichotomies can generate is $\binom{N+1}{1} = N+1$

$\Rightarrow \mathcal{H}(N) = N+1$

(2) Positive intervals: $\mathcal{H}$ consists of all hypotheses that return +1 with some interval and -1 otherwise. Each hypothesis is specified by the two end values of that interval. The illustration is as follows,

N points can separate the one-dimensional input space into N+1 intervals

to determine a single hypothesis, we need to determine the two end values

① when 2 end values are in same interval, there is only 1 kind of dichotomy:
all numbers are possitive

②: when 2 end values are in different intervals,
h can generate one dichotomy for any different combinations of end value intervals
So total number of dichotomies: $\binom{N+1}{2} = \frac{N(N+1)}{2}$

$\Rightarrow \mathcal{H}(N) = 1 + \frac{1}{2}N(N+1)$