1. Which of the following problems are best suited for machine learning?
   (i) Classifying numbers into primes and non-primes
   (ii) Detecting potential fraud in credit card charges
   (iii) Determining the time it would take a falling object to hit the ground
   (iv) Determining the optimal cycle for traffic lights in a busy intersection
   (v) Determining the age at which a particular medical test is recommended

Ans:
(ii), (iv), (v)

2. For Questions 2-5, identify the best type of learning that can be used to solve each task below.
   Play chess better by practicing different strategies and receive outcome as feedback.

Ans:
implicit y_n by goodness($\widetilde{y_n}$) → reinforcement learning

3. Categorize books into groups without pre-defined topics.

Ans:
no y_n → unsupervised learning

4. Recognize whether there is a face in the picture by a thousand face pictures and ten thousand nonface pictures.

Ans:
all y_n → supervised learning

5. Selectively schedule experiments on mice to quickly evaluate the potential of cancer medicines.

Ans:
query the y_n of the chosen x_n → active learning

6. Question 6-8 are about Off-Training-Set error.

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+L}\}$ and $\mathcal{Y} = \{-1, +1\}$ (binary classification). Here the set of training examples is $\mathcal{D} = \left\{(\mathbf{x}_n, y_n)\right\}_{n=1}^{N}$, where $y_n \in \mathcal{Y}$, and the set of test inputs is $\left\{\mathbf{x}_{N+\ell}\right\}_{\ell=1}^{L}$. The Off-Training-Set error (OTS) with respect to an underlying target f and a hypothesis g is

$$E_{OTS}(g, f) = \frac{1}{L} \sum_{\ell=1}^{L} [\![g(\mathbf{x}_{N+\ell}) \neq f(\mathbf{x}_{N+\ell})]\!].$$

Consider $f(\mathbf{x}) = +1$ for all $\mathbf{x}$ and

$$g(\mathbf{x}) = \begin{cases} +1, & \text{for } \mathbf{x} = \mathbf{x}_k \text{ and } k \text{ is odd and } 1 \leq k \leq N+L \\ -1, & \text{otherwise} \end{cases}.$$

$E_{OTS}(g, f) =$? (Please note the difference between floor and ceiling functions in the choices)

Ans:

#{error of all} = $\left\lfloor \frac{N+L}{2} \right\rfloor$, #{error of training examples} = $\left\lfloor \frac{N}{2} \right\rfloor$. → $\frac{1}{L} \times \left(\lfloor \frac{N+L}{2} \rfloor - \lfloor \frac{N}{2} \rfloor\right)$

7.

We say that a target function f can "generate" $\mathcal{D}$ in a noiseless setting if $f(\mathbf{x}_n) = y_n$ for all $(\mathbf{x}_n, y_n) \in \mathcal{D}$.

For all possible $f \colon \mathcal{X} \to \mathcal{Y}$, how many of them can generate $\mathcal{D}$ in a noiseless setting?

Note that we call two functions f_1 and f_2 the same if $f_1(\mathbf{x}) = f_2(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$.

Ans:

The outputs of training examples are fixed. → 2^L

8.

A determistic algorithm $\mathcal{A}$ is defined as a procedure that takes $\mathcal{D}$ as an input, and outputs a hypothesis g. For any two deterministic algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, if all those f that can "generate" $\mathcal{D}$ in a noiseless setting are equally likely in probability,

Ans:

$$\mathbb{E}_f\left\{E_{OTS}(\mathcal{A}_1(\mathcal{D}), f)\right\} = \mathbb{E}_f\left\{E_{OTS}(\mathcal{A}_2(\mathcal{D}), f)\right\}.$$

9.

For Questions 9-12, consider the bin model introduced in class. Consider a bin with infinitely many marbles, and let μ be the fraction of orange marbles in the bin, and ν is the fraction of orange marbles in a sample of 10 marbles. If $\mu = 0.5$, what is the probability of $\nu = \mu$? Please choose the closest number.

Ans:

$$\binom{10}{5} \cdot \left(\frac{1}{2}\right)^{10} = \frac{63}{256} \approx 0.24$$

10.

If $\mu = 0.9$, what is the probability of $\nu = \mu$? Please choose the closest number.

Ans:

$$\binom{10}{1} \cdot 0.9^9 \cdot 0.1 \approx 0.39$$

11.

If $\mu = 0.9$, what is the actual probability of $\nu \leq 0.1$?

Ans:

$$\binom{10}{0} \cdot 0.1^{10} + \binom{10}{1} \cdot 0.1^9 \cdot 0.9 = 9.1 \cdot 10^{-9}$$

12.

If $\mu = 0.9$, what is the bound given by Hoeffding's Inequality for the probability of $\nu \leq 0.1$?

Ans:

$$\varepsilon = 0.8, \mathrm{N} = 10 \rightarrow 2 \cdot e^{-2 \cdot 0.8^2 \cdot 10} \approx 5.52 \cdot 10^{-6}$$

13. Questions 13-14 illustrate what happens with multiple bins using dice to indicate 6 bins. Please note that the dice is not meant to be thrown for random experiments in this problem. They are just used to bind the six faces together. The probability below only refers to drawing from the bag.
Consider four kinds of dice in a bag, with the same (super large) quantity for each kind.

A: all even numbers are colored orange, all odd numbers are colored green

B: all even numbers are colored green, all odd numbers are colored orange

C: all small (1~3) are colored orange, all large numbers (4~6) are colored green

D: all small (1~3) are colored green, all large numbers (4~6) are colored orange

If we pick 5 dice from the bag, what is the probability that we get 5 orange 1's?

Ans:

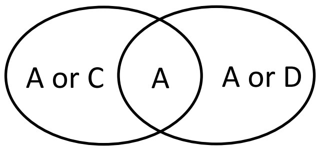
We can choose only B or C. → $\left(\frac{2}{4}\right)^5 = \frac{1}{32}$

14. If we pick 5 dice from the bag, what is the probability that we get "some number" that is purely orange?

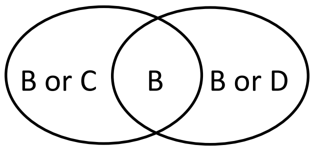
Ans:

First thing to know: if we pick A, then we can’t pick B; if C, then no D, etc.

Case 1:



Case 2:



For each case, the probability is $\left(\frac{1}{2}\right)^5 + \left(\frac{1}{2}\right)^5 - \left(\frac{1}{4}\right)^5 = \frac{63}{1024}$.

Combine two cases, and subtract the double-counted situations(only C or only D), we get $\frac{63}{1024} \cdot 2 - \left(\frac{1}{4}\right)^5 \cdot 2 = \frac{31}{256}$.

15. For Questions 15-20, you will play with PLA and pocket algorithm. First, we use an artificial data set to study PLA. The data set is in https://www.csie.ntu.edu.tw/~htlin/mooc/datasets/mlfound_math/hw1_15_train.dat

Each line of the data set contains one $(\mathbf{x}_n, y_n)$ with $\mathbf{x}_n \in \mathbb{R}^4$. The first 4 numbers of the line contains the components of $\mathbf{x}_n$ orderly, the last number is y_n.

Please initialize your algorithm with $\mathbf{w} = 0$ and take sign(0) as -1. Please always remember to add $x_0 = 1$ to each $\mathbf{x}_n$.

Implement a version of PLA by visiting examples in the naive cycle using the order of examples in the data set. Run the algorithm on the data set. What is the number of updates before the algorithm halts?

Ans:

Run PLA_original_15.py and get 45.

16.

Implement a version of PLA by visiting examples in fixed, pre-determined random cycles throughout the algorithm. Run the algorithm on the data set. Please repeat your experiment for 2000 times, each with a different random seed. What is the average number of updates before the algorithm halts?

Ans:

Run PLA_random cycles_16.py and get 40.XXXX.

17.

Implement a version of PLA by visiting examples in fixed, pre-determined random cycles throughout the algorithm, while changing the update rule to be

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta y_{n(t)} \mathbf{x}_{n(t)}$$

with $\eta = 0.5$. Note that your PLA in the previous Question corresponds to $\eta = 1$. Please repeat your experiment for 2000 times, each with a different random seed. What is the average number of updates before the algorithm halts?

Ans:

Slightly modify line 33 in PLA_random cycles_16.py and get 40.XXXX.

18.

Next, we play with the pocket algorithm. Modify your PLA in Question 16 to visit examples purely randomly, and then add the "pocket" steps to the algorithm. We will use

https://www.csie.ntu.edu.tw/~htlin/mooc/datasets/mlfound_math/hw1_18_train.dat

as the training data set $\mathcal{D}$, and

https://www.csie.ntu.edu.tw/~htlin/mooc/datasets/mlfound_math/hw1_18_test.dat

as the test set for "verifying'' the g returned by your algorithm (see lecture 4 about verifying). The sets are of the same format as the previous one. Run the pocket algorithm with a total of 50 updates on $\mathcal{D}$, and verify the performance of $\mathbf{w}_{POCKET}$ using the test set. Please repeat your experiment for 2000 times, each with a different random seed. What is the average error rate on the test set?

Ans:

Run PLA_pocket_18.py and get 0.129XXX.

19.

Modify your algorithm in Question 18 to return $\mathbf{w}_{50}$ (the PLA vector after 50 updates) instead of $\hat{\mathbf{w}}$ (the pocket vector) after 50 updates.

Run the modified algorithm on $\mathcal{D}$, and verify the performance using the test set.

Please repeat your experiment for 2000 times, each with a different random seed. What is the average error rate on the test set?

Ans:

Run PLA_pocket_19.py and get 0.265XXX.

20.

Modify your algorithm in Question 18 to run for 100 updates instead of 50, and verify the performance of $\mathbf{w}_{POCKET}$ using the test set. Please repeat your experiment for 2000 times, each with a different random seed. What is the average error rate on the test set?

Ans:

Change bound to 100 and run PLA_pocket_18.py, and get 0.114XXX.