Benjamin Ye
CS/CNS/EE 156a: Learning Systems (Fall 2023)
October 2, 2023

Homework 1

Problem	Answer
1	[d]
2	[a]
3	[d]
4	[b]
5	[c]
6	[e]
7	[b]
8	[c]
9	[b]
10	[b]

The Learning Problem

1. What types of machine learning, if any, best describe the following three scenarios?
   (i) A coin classification system is created for a vending machine. The developers obtain exact coin specifications from the U.S. Mint and derive a statistical model of the size, weight, and denomination, which the vending machine then uses to classify coins.
   (ii) Instead of calling the U.S. Mint to obtain coin information, an algorithm is presented with a large set of labeled coins. The algorithm uses this data to infer decision boundaries which the vending machine then uses to classify its coins.
   (iii) A computer develops a strategy for playing tic-tac-toe by playing repeatedly and adjusting its strategy by penalizing moves that eventually lead to losing.

   Answer: [d] (i) not learning, (ii) supervised learning, (iii) reinforcement learning

   (i) is not an example of machine learning because the statistical model was designed using the exact coin specifications and was not determined by using any data, such as the sizes and masses of the coins to be classified.

   (ii) is an example of supervised learning because the algorithm learns from data that specifies what the correct output (coin label) should be for a given input (coin information).

   (iii) is an example of reinforcement learning because the strategy is optimized using data that does not explicitly categorize any moves as "good" but nevertheless rewards those that eventually lead to a winning outcome by assigning them positive grades or scores.

2. 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.

   Answer: [a] (ii) and (iv)

   (i) and (iii) can be solved using conventional formulas and methods, such as the modulus operator and the equations of motion, respectively, so machine learning is not necessary.

   (ii) is suited for machine learning because there is no function that can detect fraudulent transactions, but a pattern can be found using transaction data (amount, location, chargeback rate, etc.).

   Similarly, (iv) is suited for machine learning because there is no function that can optimally time traffic light cycles, but a pattern can be found using traffic data (number of cars, wait time, etc.).

Bins and Marbles

3. We have 2 opaque bags, each containing 2 balls. One bag has 2 black balls and the other has a black ball and a white ball. You pick a bag at random and then pick one of the balls in that bag at random. When you look at the ball, it is black. You now pick the second ball from that same bag. What is the probability that this ball is also black?

   Answer: [d] 2/3

   From logical reasoning, once it has been established the first ball is black, there are only three possible outcomes:

   1. bag 1: black ball then white ball,
   2. bag 2: first black ball then second black ball, or
   3. bag 2: second black ball then first black ball.

   Out of the three outcomes, only the second and third gives a black ball as the second pick, so the probability is 2/3.

   A more rigorous approach involves the definition of conditional probability:

   $$P(A \cap B) = P(A|B)P(B)$$

   Let event A be picking a black ball in the second attempt and B be picking a black ball in the first attempt. There are four possible outcomes:

   1. bag 1: black ball then white ball,
   2. bag 1: white ball then black ball,
   3. bag 2: first black ball then second black ball, or
   4. bag 2: second black ball then first black ball.

   The probability of B occurring is

   $$P(B) = \frac{3}{4}$$

   because there are four balls with equal chance of being selected and three of them are black. The probability of both A and B occurring is

   $$P(A \cap B) = \frac{1}{2}$$

   because out of the four outcomes, only the latter two satisfy the criteria.

   Therefore, the probability that the second ball is black given that the first ball is black is

   $$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/2}{3/4} = \frac{2}{3}$$

Consider a sample of 10 marbles drawn from a bin containing red and green marbles. The probability that any marble we draw is red is $\mu = 0.55$ (independently, with replacement). We address the probability of getting no red marbles ($\nu = 0$) in the following cases:

4. We draw only one such sample. Compute the probability that $\nu = 0$. The closest answer (|your answer − given option| closest to 0) is…

   Answer: [b] 3.405×10^{-4}

   $\nu = 0$ is attainable only when none of the marbles are red (or all the marbles are green). Using a binomial distribution with probability $p = 0.55$, sample size $n = 10$, and $k = 0$ successes, the probability is

   $$P(\nu = 0) = \binom{10}{0}(0.55)^0(1-0.55)^{10-0} = (0.45)^{10} \approx 3.405 \times 10^{-4}$$

5. We draw 1,000 independent samples. Compute the probability that (at least) one of the samples has $\nu = 0$. The closest answer is…

   Answer: [c] 0.289

   The probability that none of the samples have $\nu = 0$ can be determined using a binomial distribution with probability $p = 3.405 \times 10^{-4}$ (from problem 4), sample size $n = 1{,}000$, and $k = 0$ successes:

   $$P(k = 0) = \binom{1{,}000}{0}(3.405 \times 10^{-4})^0(1 - 3.405 \times 10^{-4})^{1{,}000-0} \approx 0.999659494^{1{,}000} \approx 0.711$$

   Then, the probability that at least one of the samples has $\nu = 0$ is simply

   $$P(k > 0) = 1 - P(k = 0) \approx 0.289$$

Feasibility of Learning

Consider a Boolean target function over a three-dimensional input space $\mathcal{X} = \{0, 1\}^3$ (instead of our ± 1 binary convention, we use 0, 1 here since it is standard for Boolean functions). We are given a data set $\mathcal{D}$ of five examples represented in the table below, where $y_n = f(\mathbf{x}_n)$ for $n = 1, 2, 3, 4, 5$.

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

Note that in this simple Boolean case, we can enumerate the entire input space (since there are only $2^3 = 8$ distinct input vectors), and we can enumerate the set of all possible target functions (there are only $2^{2^3} = 256$ distinct Boolean functions on three Boolean inputs).

Let us look at the problem of learning f. Since f is unknown except inside $\mathcal{D}$, any function that agrees with $\mathcal{D}$ could conceivably be f. Since there are only three points in $\mathcal{X}$ outside $\mathcal{D}$, there are only $2^3 = 8$ such functions.

The remaining points in $\mathcal{X}$ which are not in $\mathcal{D}$ are 101, 110, and 111. We want to determine the hypothesis that agrees the most, *on these three points*, with the possible target functions. To quantify this, count how many of the eight possible target functions agree with each hypothesis on all three points, on just two of the points, on just one point, and on none of the points. How a hypothesis agrees with the target function in sample (on $\mathcal{D}$ itself) has no bearing on its score:

Score = (# of target functions agreeing with hypothesis on all three points)] × 3
+ (# of target functions agreeing with hypothesis on exactly two points) × 2
+ (# of target functions agreeing with hypothesis on exactly one point) × 1
+ (# of target functions agreeing with hypothesis on no points) × 0

6. Which hypothesis g agrees the most with the possible target functions in terms of the above score?

Answer: [e] They are all equivalent (equal scores for g in [a] through [d]).

Purely by symmetry for binary data, there will always be one target function agreeing with the hypothesis on all three points, three target functions with two points, three target functions with one point, and one target function with no points for a total score of 12:

[a] Score = $1 \times 3 + 3 \times 2 + 3 \times 1 + 1 \times 0 = 12$

$\mathbf{x}_n$			y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0	1		1	0	0	0	0	1	1	1	1
1	1	0		1	0	0	1	1	0	0	1	1
1	1	1		1	0	1	0	1	0	1	0	1

[b] Score = $1 \times 3 + 3 \times 2 + 3 \times 1 + 1 \times 0 = 12$

$\mathbf{x}_n$			y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0	1		0	0	0	0	0	1	1	1	1
1	1	0		0	0	0	1	1	0	0	1	1
1	1	1		0	0	1	0	1	0	1	0	1

[c] Score = $1 \times 3 + 3 \times 2 + 3 \times 1 + 1 \times 0 = 12$

$\mathbf{x}_n$			y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0	1		0	0	0	0	0	1	1	1	1
1	1	0		0	0	0	1	1	0	0	1	1
1	1	1		1	0	1	0	1	0	1	0	1

[d] Score = $1 \times 3 + 3 \times 2 + 3 \times 1 + 1 \times 0 = 12$

$\mathbf{x}_n$			y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0	1		1	0	0	0	0	1	1	1	1
1	1	0		1	0	0	1	1	0	0	1	1
1	1	1		0	0	1	0	1	0	1	0	1

The Perceptron Learning Algorithm

In this problem, you will create your own target function f and data set $\mathcal{D}$ to see how the perceptron learning algorithm (PLA) works. Take $d = 2$ so you can visualize the problem and assume $\mathcal{X} = [-1, 1] \times [-1, 1]$ with uniform probability of picking each $\mathbf{x} \in \mathcal{X}$.

In each run, choose a random line in the plane as your target function f (do this by taking two random, uniformly distributed points in $[-1, 1] \times [-1, 1]$ and taking the line passing through them), where one side of the line maps to $+1$ and the other maps to -1. Choose the inputs $\mathbf{x}_n$ of the data set as random points (uniformly in $\mathcal{X}$) and evaluate the target function on each $\mathbf{x}_n$ to get the corresponding output y_n.

Now, in each run, use the PLA to find g. Start the PLA with the weight vector $\mathbf{w}$ being all zeros (consider $\text{sgn}(0) = 0$, so all points are initially misclassified), and at each iteration have the algorithm choose a point randomly from the set of misclassified points. We are interested in two quantities: the number of iterations that PLA takes to converge to g, and the disagreement between f and g which is $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ (the probability that f and g will disagree on their classification of a random point). You can either calculate the probability exactly, or approximate it by generating a sufficiently large, separate set of points to estimate it.

To get a reliable estimate for these two quantities, you should repeat the experiment for 1,000 runs (each run as specified above) and take the average over these runs.

7. Take $N = 10$. How many iterations does it take on average for the PLA to converge for $N = 10$ training points? Pick the value closest to your results.

   Answer: [b] 15

8. Which of the following is closest to $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ for $N = 10$?

   Answer: [c] 0.1

9. Now, try $N = 100$. How many iterations does it take on average for the PLA to converge for $N = 100$ training points? Pick the value closest to your results.

   Answer: [b] 100

10. Which of the following is closest to $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ for $N = 100$?

    Answer: [b] 0.01

For the perceptron, the target function is a simple line $f(\mathbf{x}) = mx + b$, where x is the x-coordinate of point $\mathbf{x}_n$ in data set $\mathcal{D}$, and the misclassification rate is estimated by using a test data set with $N = 1{,}000$. The sample output from the program used to answer problems 7–10 is

```
[HW1 P7-10]
PLA statistics over 1,000 runs:
  N=10, iters=10, prob=0.107
  N=100, iters=114, prob=0.014
```

(The Python 3 source code is available on the following page.)

```python
import numpy as np

def target_function_random_line(*, rng=None, seed=None):
    if rng is None:
        rng = np.random.default_rng(seed)
    line = rng.uniform(-1, 1, (2, 2))
    return lambda x: np.sign(
        x[:, 2] - line[0, 1]
        - np.divide(*(line[1] - line[0])[::-1]) * (x[:, 1] - line[0, 0])
    )

def generate_data(N, f, d=2, lb=-1.0, ub=1.0, *, rng=None, seed=None):
    if rng is None:
        rng = np.random.default_rng(seed)
    x = np.hstack((np.ones((N, 1)), rng.uniform(lb, ub, (N, d))))
    return x, f(x)

def validate_binary(w, x, y):
    return np.count_nonzero(np.sign(x @ w) != y, axis=0) / x.shape[0]

def perceptron(N, f, *, N_test=1_000, rng=None, seed=None):
    if rng is None:
        rng = np.random.default_rng(seed)
    x, y = generate_data(N, f, rng=rng)
    w = np.zeros(x.shape[1], dtype=float)
    iters = 0
    while True:
        wrong = np.argwhere(np.sign(x @ w) != y)[:, 0]
        if wrong.size == 0:
            break
        i = np.random.choice(wrong)
        w += y[i] * x[i]
        iters += 1
    return iters, validate_binary(w, *generate_data(N_test, f, rng=rng))

if __name__ == "__main__":
    rng = np.random.default_rng()
    n_runs = 1_000
    print(f"\n[HW1 P7-10]\nPLA statistics over {n_runs:,} runs:")
    for N in (10, 100):
        iters, prob = np.mean(
            [perceptron(N, target_function_random_line(rng=rng), rng=rng)
             for _ in range(n_runs)],
            axis=0
        )
        print(f"  {N=:,}, {iters=:,.0f}, {prob=:.3f}")
```