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?
2. 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.
3. 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.
4. 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.

1. Which of the following problems are best suited for machine learning?
2. Classifying numbers into primes and non-primes.
3. Detecting potential fraud in credit card charges.
4. Determining the time it would take a falling object to hit the ground.
5. 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**

1. 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]**

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 .

A more rigorous approach involves the definition of conditional probability:

Let event be picking a black ball in the second attempt and 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 occurring is

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

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

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 (independently, with replacement). We address the probability of getting no red marbles () in the following cases:

1. We draw only one such sample. Compute the probability that . The closest answer ( closest to 0) is…

**Answer: [b]**

is attainable only when none of the marbles are red (or all the marbles are green). Using a binomial distribution with probability , sample size , and successes, the probability is

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

**Answer: [c] 0.289**

The probability that none of the samples have can be determined using a binomial distribution with probability (from Problem 4), sample size , and successes:

Then, the probability that at least one of the samples has is simply

**Feasibility of Learning**

Consider a Boolean target function over a three-dimensional input space (instead of our binary convention, we use here since it is standard for Boolean functions). We are given a data set of five examples represented in the table below, where for .

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
| 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 distinct input vectors), and we can enumerate the set of all possible target functions (there are only distinct Boolean functions on three Boolean inputs).

Let us look at the problem of learning . Since is unknown except inside , any function that agrees with could conceivably be . Since there are only three points in outside , there are only such functions.

The remaining points in which are not in are , , and . 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 itself) has no bearing on its score:

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

**Answer: [e] They are all equivalent (equal scores for 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 :

[a]

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  |  |  |  |  |  |  |  |  |  |
| 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]

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  |  |  |  |  |  |  |  |  |  |
| 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]

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  |  |  |  |  |  |  |  |  |  |
| 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]

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  |  |  |  |  |  |  |  |  |  |
| 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 and data set to see how the perceptron learning algorithm (PLA) works. Take so you can visualize the problem and assume with uniform probability of picking each .

In each run, choose a random line in the plane as your target function (do this by taking two random, uniformly distributed points in and taking the line passing through them), where one side of the line maps to and the other maps to . Choose the inputs of the data set as random points (uniformly in ) and evaluate the target function on each to get the corresponding output .

Now, in each run, use the PLA to find . Start the PLA with the weight vector being all zeros (consider , 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 , and the disagreement between and which is (the probability that and 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.

1. Take . How many iterations does it take on average for the PLA to converge for training points? Pick the value closest to your results.

**Answer: [b] 15**

1. Which of the following is closest to for ?

**Answer: [c] 0.1**

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

**Answer: [b] 100**

1. Which of the following is closest to for ?

**Answer: [b] 0.01**

For the perceptron, the target function is a simple line , where is the -coordinate of point in data set , and the misclassification rate is estimated by using a new data set with points.

(The sample program output and Python 3 source code are on the following pages. )

**Sample program output**

[Homework 1 Problems 7–10]

Perceptron learning algorithm (1,000 runs):

number of points number of iterations misclassification rate

10 **9** **0.105178**

100 **104** **0.013218**

**Python 3 source code**

import numpy as np

import pandas as pd

class Perceptron:

    def \_\_init\_\_(self, w=None, \*, vf=None):

        self.set\_parameters(w, vf=vf)

    def get\_error(self, x, y):

        if self.vf is not None and self.w is not None:

            return self.vf(self.w, x, y)

    def set\_parameters(self, w=None, \*, vf=None, update=False) -> None:

        if update:

            self.vf = vf or self.vf

            self.\_w = self.\_w if w is None else w

        else:

            self.vf = vf

            self.\_w = w

    def train(self, x, y):

        self.iters = 0

        self.w = (np.zeros(x.shape[1], dtype=float) if self.\_w is None

                  else self.\_w)

        while True:

            wrong = np.argwhere(np.sign(x @ self.w) != y)[:, 0]

            if wrong.size == 0:

                break

            index = np.random.choice(wrong)

            self.w += y[index] \* x[index]

            self.iters += 1

        if self.vf:

            return self.vf(self.w, x, y)

def target\_function\_random\_line(x=None, \*, rng=None, seed=None):

    if rng is None:

        rng = np.random.default\_rng(seed)

    line = rng.uniform(-1, 1, (2, 2))

    f = lambda x: np.sign(

        x[:, -1] - line[0, 1]

        - np.divide(\*(line[1] - line[0])[::-1]) \* (x[:, -2] - line[0, 0])

    )

    return f if x is None else f(x)

def generate\_data(

        N, f, d=2, lb=-1.0, ub=1.0, \*, bias=False, rng=None, seed=None):

    if rng is None:

        rng = np.random.default\_rng(seed)

    x = rng.uniform(lb, ub, (N, d))

    if bias:

        x = np.hstack((np.ones((N, 1)), x))

    return x, f(x)

def validate\_binary(w, x, y):

    return np.count\_nonzero(np.sign(x @ w) != y, axis=0) / x.shape[0]

if \_\_name\_\_ == "\_\_main\_\_":

    rng = np.random.default\_rng()

    N\_runs = 1\_000

    pla = Perceptron(vf=validate\_binary)

    columns = ["number of points", "number of iterations",

               "misclassification rate"]

    df = pd.DataFrame(columns=columns)

    for N\_train in (10, 100):

        N\_test = 9 \* N\_train

        counters = np.zeros(2, dtype=float)

        for \_ in range(N\_runs):

            f = target\_function\_random\_line(rng=rng)

            pla.train(\*generate\_data(N\_train, f, bias=True, rng=rng))

            counters += (

                pla.iters,

                pla.get\_error(\*generate\_data(N\_test, f, bias=True, rng=rng))

            )

        df.loc[len(df)] = N\_train, \*(counters / N\_runs)

    print("\n[Homework 1 Problems 7–10]\n"

          f"Perceptron learning algorithm ({N\_runs:,} runs):\n",

          df.to\_string(index=False,

                       formatters={c: "{:.0f}".format for c in columns[:2]}),

          sep="")