CS 156a Set 1

Cason Shepard

January 18, 2023

problem 1

The first scenario is not an example of learning. This is because the coin vendors are receiving exact measurements of the coins size, weight, and denomination. Thus, machine learning is not useful here.

The second scenario is an example of supervised learning. This example is 'supervised' because the coins are labeled with their specific value, but the boundaries between coin classifications are not exact, and require machine learning to figure out.

The third scenario is an example of reinforcement learning. This is because after each input, a grade for the output (or outcome in this case) is returned. In this example, a poor output would be a losing game, while a preferred output would be a winning game. Thus, through reinforcement of positive outcomes, the machine learns the best strategies.

The answer is [d]

problem 2

(i) For classifying numbers as prime/non-prime, machine learning is not necessary as this is a simple computation.

(ii) **For determining potential fraud in credit card charges, machine learning could be useful.** This is because there are many factors that may point to a fraudulent charge, but none are clear indicators. For example: amount of charges, date/time of charges, location charges were applied, etc. Furthermore, there is a lot of historical data on fraudulent credit card charges that would be useful through machine learning.

(iii) For this instance, simple physics formulas would suffice to make these calculations. Machine learning would not be a suitable way to approach this problem.

(iv) **This problem would also be very well suited for Machine Learning.** At a busy intersection, there would be a lot of available data on the number/frequency of cars going any given direction. Reinforcement learning could also be useful if there was a system efficiency rating that could be used to give real-time feedback to the traffic light cycle.

The answer is [a]

problem 3

There are four possible combinations of balls that can be drawn if we are restricted to drawing the second ball from the same bag as the first ball. Once a black ball has been drawn, there are 3 different possible second-ball-options. Let 0 denote a black ball and 1 denote a white ball such that 01 represents drawing a black ball and a white ball immediately after. These possibility are 00, 00, 01. In only one case a white ball is selected second after a black ball. Thus, the probability is 2/3.

The answer is [d]

problem 4

Assuming that we are replacing each marble after drawing, the probability of drawing a non-red marble is .45. Thus, after 10 marbles are drawn consecutively, the chance that none of them are red is:

$$.45^{10} = 3.405 * 10^{-4}$$

The answer is [b]

problem 5

To find the probability of not drawing a red marble in any single 10-marble draw over 1000 trials will be the probability found in problem 4 multiplied by 1000.

$$3.405 * 10^{-4} * 1000 = .3405$$

Since .3405 is closest to b, **the answer is [c].**

problem 6

We cannot figure out how well the hypothesis g agrees on the three points outside the data set, as each hypothesis is equally likely to agree with the target function outside the training data. Thus, they are equivalent.

The answer is [e]

The following algorithm was used to answer questions 7-10

```
import matplotlib.pyplot as plt
import numpy as np
from random import uniform
import random
def PLA_func():

    #SETUP
    x_1 = uniform(-1, 1)
    y_1 = uniform(-1, 1)

    x_2 = uniform(-1, 1)
    y_2 = uniform(-1, 1)

    a, b = np.polyfit([x_1, y_1], [x_2, y_2], 1)

    def f(x): #target function
        return (a*x + b)

    def sign(x):
        if x == 0:
            return 0
        elif x > 0:
            return 1
        else:
            return -1

    def reclassify(w, PLA, points):
        for idx in range(0, len(PLA)):
            if np.dot(points[idx], w) > 0:
                PLA[idx] = 1
            elif np.dot(points[idx], w) < 0:
                PLA[idx] = -1
            else:
                PLA[idx] = 0

    N = 100
    y_target = [0] * N
    point_list = [] * N
    classification_list = [0] * N
    PLA = [0]*N

    for i in range(0,N):
        point_list.append([1, uniform(-1, 1), uniform(-1, 1)])
        y_target[i] = f(point_list[i][1])

    y_data = [x[2] for x in point_list]

    for i in range(0,N):
        if y_data[i] > y_target[i]:
            classification_list[i] = 1
        else:
            classification_list[i] = -1

    w = [0,0,0] # g(x)
    itter_count = 0

    while PLA != classification_list:
        itter_count = itter_count + 1
        wrong = []
        for p in range(0, N):
            if classification_list[p] != PLA[p]:
                wrong.append(p)
        index_point = random.choice(wrong)
```

```
        random_point_x = point_list[index_point]
        #grab sign of correct classification
        temp_sign = classification_list[index_point]
        # unpack and repack weight tuple
        w = [w[0] + temp_sign*random_point_x[0], w[1] + temp_sign*random_point_x[1], w[2] + temp_sign*ra
        reclassify(w, PLA, point_list)

    large_tuple_list = []
    large_y_target = [0] *1000
    for i in range(0, 1000):
        large_tuple_list.append([1, uniform(-1, 1), uniform(-1, 1)])
        large_y_target[i] = f(large_tuple_list[i][1])

    large_y = [x[2] for x in large_tuple_list]

    large_class = [0]*1000
    large_PLA = [0]*1000

    for i in range(0, 1000):
        if large_y[i] > large_y_target[i]:
            large_class[i] = 1
        else:
            large_class[i] = -1

    reclassify(w, large_PLA, large_tuple_list)
    count = 0
    for i in range(0, 1000):
        if large_PLA[i] != large_class[i]:
            count = count + 1

    return count/1000
```

problem 7

After running this algorithm with $N = 10$ for 1000 trials, the PLA converges to the target function in an average of 29.6 iterations, which is closest to 15.

The answer is [b]

problem 8

After running this algorithm with $N = 10$ for 1000 trials, the average probability that $f(x) \neq g(x)$ is 0.25, which is closest to 0.1.

The answer is [c]

problem 9

After running this algorithm with $N = 100$ for 1000 trials, the PLA converges to the target function in an average of 111.65 iterations, which is closest to 100.

The answer is [b]

problem 10

After running this algorithm with $N = 100$ for 1000 trials, the average probability that $f(x) \neq g(x)$ is 0.0138, which is closest to 0.01.

The answer is [b]