HW1

December 8, 2023

0.1 CS156A Homework 2

0.2 Wilson Duan

0.2.1 Problem 1.

i) This scenario is not learning because the exact coin specifications are given, meaning the vending machine already has a prescribed set of rules to classify the coins.

ii) This scenario is supervised learning because the algorithm is given labels in order to infer decision boundaries.

iii) This scenario is reinforcement learning because the computer is learning Tic-Tac-Toe by responding to rewards and punishments.

Therefore, the answer is **d)**.

0.2.2 Problem 2.

i) Classifying numbers into primes and non-primes is not suited for Machine Learning because it can be solved with a simple for loop.

ii) Detecting potential fraud in credit card charges is well suited for Machine Learning because it is not a straightforward task; there are many factors that contribute to fraud, which can be learned by a Machine Learning model. There is an expansive amount of data relating to credit card fraud that can be provided to the Machine Learning model for it to learn from.

iii) Determining the time it would take a falling object to hit the ground is not suited for Machine Learning because it is solved easily by kinematic equations.

iv) Determining the optimal cycle for traffic lights is well suited for Machine Learning because it is a complex task with many parameters that a model can learn by identifying patterns.

Therefore, the answer is **a)**.

0.2.3 Problem 3.

We denote bag 1 as the bag with two black balls, and bag 2 as the bag with 1 black ball and one white ball.

$P(\text{second ball is black} \mid \text{first ball is black}) = \frac{P(\text{second ball is black} \cap \text{first ball is black})}{P(\text{first ball is black})}$.

$P(\text{second ball is black } \cap \text{ first ball is black}) = 1/2$ because this happens only if we picked bag 1.

$P(\text{first ball is black}) = 1/2 * P(\text{choosing bag 2}) + 1 * P(\text{choosing bag 1}) = 1/2 * 1/2 + 1/2 = 3/4$

$P(\text{second ball is black} \mid \text{first ball is black}) = \frac{1/2}{3/4} = 2/3$

Therefore, the answer is **d)**.

0.2.4 Problem 4.

$P(v = 0) = (1 - \mu)^{10} = 0.45^{10} = 3.405 \times 10^{-4}$. Therefore, the answer is **b)**.

0.2.5 Problem 5.

$P(\text{at least one sample has } v = 0) = 1 - P(\text{no sample has } v = 0) = 1 - (1 - 3.405 \times 10^{-4})^{1000} = 0.289$. Therefore, the answer is **c)**.

0.2.6 Problem 6.

a) g returns 1 for all three points. Score = (# of functions that put 1 on all three points) * 3 + (# of functions that put 1 on two points) * 2 + (# of functions that put 1 on one point) * 1 + (# of functions that put 1 on no points) * 0 = (1) * 3 + (3) * 2 + (3) * 1 + (1) * 0 = 12

b) g returns 0 for all three points. Score = (# of functions that put 0 on all three points) * 3 + (# of functions that put 0 on two points) * 2 + (# of functions that put 0 on one point) * 1 + (# of functions that put 0 on no points) * 0 = (1) * 3 + (3) * 2 + (3) * 1 + (1) * 0 = 12

c) g is the XOR function. $g(101) = 0, g(110) = 0, g(111) = 1$. Score = (# of functions that agree on all 3 points) * 3 + (# of functions that agree on 2 points) * 2 + (# of functions that agree on 1 point) * 1 + (# of functions that agree on 0 points) = (1) * 3 + (3) * 2 + (3) * 1 + (1) * 0 = 12

d) g is the opposite of the XOR function. $g(101) = 1, g(110) = 1, g(111) = 0$. Score = (# of functions that agree on all 3 points) * 3 + (# of functions that agree on 2 points) * 2 + (# of functions that agree on 1 point) * 1 + (# of functions that agree on 0 points) = (1) * 3 + (3) * 2 + (3) * 1 + (1) * 0 = 12

Since all of the hypotheses have the same score above, they are all equivalent and the answer is **e)**.

0.2.7 Problem 7.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
random.seed(123)
```

[2]:
```
# Define a set of helper functions
def random_point():
    x = random.random() * 2 - 1
    y = random.random() * 2 - 1
    return (x, y)

def random_line():
    x1, y1 = random_point()
    x2, y2 = random_point()
```

```python
    slope = (y2 - y1) / (x2 - x1)
    intercept = y1 - slope * x1
    return (slope, intercept)

def evaluate_point(slope, intercept, x, y):
    if (slope * x + intercept > y):
        return -1
    return 1

def predict(weights, x, y):
    return np.sign(weights[0] * x + weights[1] * y + weights[2])
```

[3]:

```python
def create_dataset(n, slope, intercept):
    X = []
    y = []
    for i in range(n):
        a, b = random_point()
        X.append((a, b))
        y.append(evaluate_point(slope, intercept, a, b))
    return X, y
```

[4]:

```python
def simulate_PLA(n):
    slope, intercept = random_line()
    X, y = create_dataset(n, slope, intercept)

    weights = np.array([0.0, 0.0, 0.0])
    iterations = 0

    # calibrate weights
    while True:
        misclassified_points = []
        # populate misclassified points
        for ((a, b), label) in zip(X, y):
            prediction = predict(weights, a, b)
            if (prediction != label):
                misclassified_points.append((a, b, label))

        # check for convergence
        if (len(misclassified_points) == 0):
            break
        else:
            a, b, label = random.choice(misclassified_points)
            weights += label * np.array([a, b, 1])
            iterations += 1

    # evaluate performance
```

```python
    incorrect = 0.0
    for i in range(1000):
        a, b = random_point()
        prediction = predict(weights, a, b)
        label = evaluate_point(slope, intercept, a, b)
        incorrect += (int)(prediction != label)
    disagreement = incorrect / 1000

    return iterations, disagreement
```

[5]:
```python
# simulate 1000 runs and take average
N = 10
iteration_sum = 0
disagreement_sum = 0
for i in range(1000):
    iterations, disagreement = simulate_PLA(N)
    iteration_sum += iterations
    disagreement_sum += disagreement

print(f"Average # of iterations to converge for N = {N}: ", iteration_sum /␣
 ↪1000)
print(f"Average P[f(x) != g(x)] for N = {N}: ", disagreement_sum / 1000)
```

```
Average # of iterations to converge for N = 10:  11.454
Average P[f(x) != g(x)] for N = 10:  0.10520700000000005
```

According to the code output above, the answer is **b)**.

0.2.8 Problem 8.

According to the code output above, the answer is **c)**.

0.2.9 Problem 9.

[6]:
```python
# repeat but with N = 100
N = 100
iteration_sum = 0
disagreement_sum = 0
for i in range(1000):
    iterations, disagreement = simulate_PLA(N)
    iteration_sum += iterations
    disagreement_sum += disagreement

print(f"Average # of iterations to converge for N = {N}: ", iteration_sum /␣
 ↪1000)
print(f"Average P[f(x) != g(x)] for N = {N}: ", disagreement_sum / 1000)
```

```
Average # of iterations to converge for N = 100:  104.068
Average P[f(x) != g(x)] for N = 100:  0.013332999999999963
```

According to the code output above, the answer is **b)**.

0.2.10 Problem 10.

According to the code output above, the answer is **b)**.