# CaltechX: Learning From Data: Homework #1

Due on Monday, October 2, 2017

*Yaser Abu-Mostafa*

**Andrew Watson**

# Contents

# The Learning Problem

## 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.

[**a**] (i) Supervised Learning, (ii) Unsupervised Learning, (iii) Reinforcement Learning

[**b**] (i) Supervised Learning, (ii) Not learning, (iii) Unsupervised learning

[**c**] (i) Not learning, (ii) Reinforcement Learning, (iii) Supervised Learning

[**d**] (i) Not learning, (ii) Supervised Learning, (iii) Reinforcement Learning

[**e**] (i) Supervised Learning, (ii) Reinforcement Learning, (iii) Unsupervised learning

- (i) is not learning, because the classification is explicitly using information about the distribution of each coin, not features learned from data.
- (ii) is supervised learning, because you are only classifying coins based on data that is given, and the data is labeled.
- (iii) is reinforcement learning, because the classification has *some* output (namely the outcome of a sequence of moves), but it doesn't have the output of each individual move.

Therefore the answer is [**d**].

## Problem 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.

[**a**] (ii) and (iv)

[**b**] (i) and (ii)

[**c**] (i), (ii), and (iii)

[**d**] (iii)

Problem 2 continued on next page...

[**e**] (i) and (iii)

- (i) is a well-defined mathematical problem, and therefore not suited for Machine Learning.
- (ii) is not a well-defined mathematical problem, but there is likely a pattern that can be used to detect fraud, and there is plenty of credit card data available, therefore this problem is suitable for Machine Learning.
- (iii) is a physical problem for which there is a simple and accurate mathematical model, therefore this is not suitable for Machine Learning
- (iv) is a problem which may be well defined mathematically, but one whose analytic solution is likely intractable, therefore this is a good problem for Machine Learning.

Therefore the answer is [**a**].

# Bins and Marbles

## Problem 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?

[**a**] 1/4

[**b**] 1/3

[**c**] 1/2

[**d**] 2/3

[**e**] 3/4

There are three possible outcomes from drawing two marbles:

| Order of drawing | Probability |
|---|---|
| black, then black | 1/2 |
| black, then white | 1/4 |
| white, then black | 1/4 |

To find the probability that you draw a black ball on your second draw given that you drew a black ball on the first draw, we can use the formula for conditional probability:

$$\mathbb{P}[\text{drawing black second} \mid \text{drew black first}] = \frac{\mathbb{P}[\text{drawing two black marbles}]}{\mathbb{P}[\text{drawing black first}]} = \frac{1/2}{3/4} = 2/3.$$

Therefore the answer is [**d**].

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:

### Problem 4

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

[**a**] $7.331 \times 10^{-6}$

[**b**] $3.405 \times 10^{-4}$

[**c**] 0.289

[**d**] 0.450

[**e**] 0.550

The probability of drawing no red marbles in a sample of 10 marbles is $(1-\mu)^10 \approx 0.00034$. Therefore the answer is [**b**].

### Problem 5

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

[**a**] $7.331 \times 10^{-6}$

[**b**] $3.405 \times 10^{-4}$

[**c**] 0.289

[**d**] 0.450

[**e**] 0.550

In the previous problem, we found that $\mathbb{P}[\nu = 0] = 3.405 \times 10^{-4}$. Note that because the samples are independent and identically distributed

$$\begin{aligned}\mathbb{P}[\text{there is at least one } i = 1, \dots 1000 \text{ such that } \nu_i = 0] &= 1 - \mathbb{P}[\nu_i > 0 \text{ for every } i = 1, \dots 1000] \\ &= 1 - \prod_{i=1}^{1000} \mathbb{P}[\nu_i > 0] = 1 - \prod_{i=1}^{1000} (1 - \mathbb{P}[\nu_i = 0]) = 1 - (1 - \mathbb{P}[\nu = 0])^{1000} \approx 0.289.\end{aligned}$$

Therefore the answer is [**c**].

## Feasibility of Learning

Consider a Boolean target function over a 3-dimensional input space $\mathcal{X} = \{0, 1\}^3$ (instead of our $\pm 1$ 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 function on 3 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 3 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 with the possible target functions. In order to quantify this, count how many of the 8 possible target functions agree with each hypothesis on all 3 points, how many agree on just 2 of the points, on just 1 point, and how many do not agree on any points. The final score for each hypothesis is computed as follows:
**Score** = (*# of target functions agreeing with hypothesis on all 3 points*)$\times 3$ + (*# of target functions agreeing with hypothesis on exactly 2 points*)$\times 2$ + (*# of target functions agreeing with hypothesis on exactly 1 point*)$\times 1$ + (*# of target functions agreeing with hypothesis on 0 points*)$\times 0$.

### Problem 6

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

[**a**] $g$ returns 1 for all three points.

[**b**] $g$ returns 0 for all three points.

[**c**] $g$ is the XOR function applied to $\mathbf{x}$, i.e., if the number of 1s in $\mathbf{x}$ is odd, $g$ returns 1; if it is even, $g$ returns 0.

[**d**] $g$ returns the opposite of the XOR function: if the number of 1s is odd, it returns 0, otherwise returns 1.

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

Below is a table listing the 8 possible target functions, and the number of target functions agreeing with each hypothesis listed above on all of the points in $\mathcal{X} - \mathcal{D}$.

| $\mathcal{X} - \mathcal{D}$ | Potential target functions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 101 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| 110 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 111 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| **Hypotheses** | **Agreement with each target** | | | | | | | | **Total Score** |
| $g_a$ | 0 | 1 | 1 | 2 | 1 | 2 | 2 | 3 | 12 |
| $g_b$ | 3 | 2 | 2 | 1 | 2 | 1 | 1 | 0 | 12 |
| $g_c$ | 2 | 3 | 1 | 2 | 1 | 2 | 0 | 1 | 12 |
| $g_d$ | 1 | 0 | 2 | 1 | 2 | 1 | 3 | 2 | 12 |

Each hypothesis has a score of 12, so the answer is [**e**].

## 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 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 Perceptron Learning Algorithm to find $g$. Start the PLA with the weight vector $\mathbf{w}$ being all zeros (consider sign(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 this probability exactly, or approximate it by generating a sufficiently large, separate set of points to estimate it.
In order to get a reliable estimate for these two quantities, you should repeat the experiment for 1000 runs (each run as specified above) and take the average over these runs.

## Problem 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 (again, 'closest' means: |your answer – given option| is closest to 0).

[**a**] 1

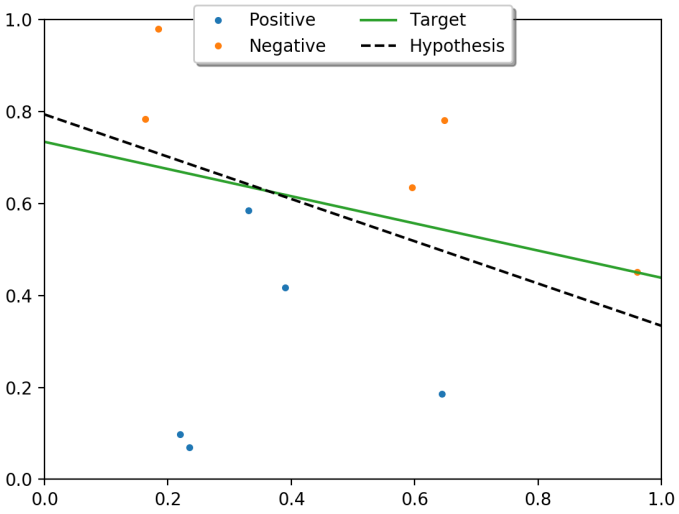
[**b**] 15

[**c**] 300

[**d**] 5000

[**e**] 10000

Below is a plot showing the target function, training data, and the PLA hypothesis for 10 points.




The perceptron code in `Homework1.py` took 22 iterations on average to converge, and had an average out-of-sample classification error of 0.110.
Therefore the answer is [**b**].

Problem 7 continued on next page...

## Problem 8

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

[**a**] 0.001

[**b**] 0.01

[**c**] 0.1

[**d**] 0.5

[**e**] 0.8

The answer is [**c**]. (See answer above)

## Problem 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.

[**a**] 50

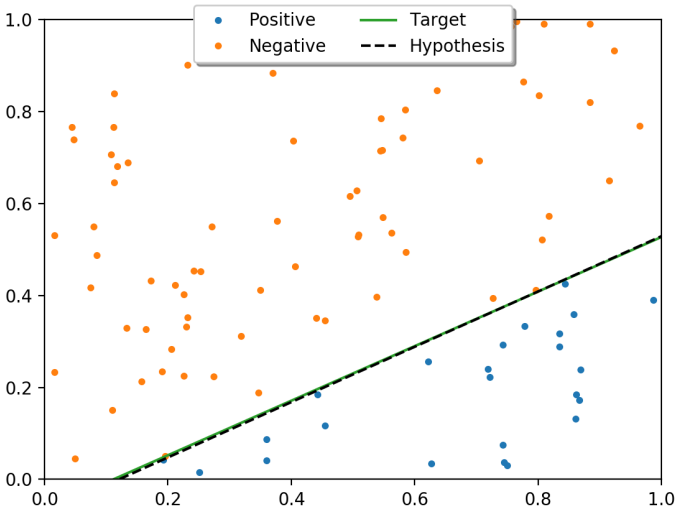
[**b**] 100

[**c**] 500

[**d**] 1000

[**e**] 5000

Below is a plot showing the target function, training data, and the PLA hypothesis for 100 points.




The perceptron code in `Homework1.py` took 179 iterations on average to converge, and had an average out-of-sample classification error of 0.013.
Therefore the answer is [**b**].

Problem 9 continued on next page...

## Problem 10

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

[**a**] 0.001

[**b**] 0.01

[**c**] 0.1

[**d**] 0.5

[**e**] 0.8

The answer is [**b**]. (See answer above)