2019 ML foundation HW 1

b07902064 資工二 蔡銘軒

October 31, 2019

Problem 1

Problem 2

One application of semi-supervised learning is facial expression recognition. We may associate certain facial features with an emotion, such as a smile lift and squinting eyes as features of being happy, and design a mathematical model to compute whether there are enough features to represent certain emotions. Although there are a huge amount of face images available, labeling each of them is tedious and laborious, and this is where semi-supervised learning is applicable. We can label a small amount of the images, combined with the unlabeled images as input data, and expect the learning algorithm to produce the emotions present in the images. In fact, there are several researchers and professors in different universities in USA working on this particular topic.

Problem 3

The concept of "no free lunch" states that, for any $\mathcal{A}(\mathcal{D})$, we can't guarantee its performance outside $\mathcal{D}$. We first note that there are 2^L different f that can generate $\mathcal{D}$, and each of them are equally likely in probability. With different f, $\mathcal{A}(\mathcal{D})$ can make any number of mistakes, ranging from 0 to L, outside $\mathcal{D}$. So we consider the probability of $\mathcal{A}(\mathcal{D})$ making 0 error, 1 error, 2 errors, ..., L errors, and we can calculate the expectation value as:

$$\mathbb{E}_f\Big\{E_{OTS}\big(\mathcal{A}(\mathcal{D}),f\big)\Big\} = \frac{1}{L}\left(\frac{0\cdot C_0^L + 1\cdot C_1^L + 2\cdot C_2^L + \cdots + L\cdot C_L^L}{2^L}\right) \tag{1}$$

$$= \frac{1}{L}\frac{\sum_{k=0}^{L} k\cdot C_k^L}{2^L} = \frac{1}{L}\frac{\sum_{k=0}^{L} k\cdot \frac{L!}{k!(L-k)!}}{2^L} \tag{2}$$

$$= \frac{1}{L}\frac{\sum_{k=1}^{L} L\cdot \frac{(L-1)!}{(k-1)!(L-k)!}}{2^L} = \frac{1}{L}\frac{L\cdot\sum_{k=1}^{L} C_{k-1}^{L-1}}{2^L} \tag{3}$$

$$= \frac{\sum_{k=0}^{L-1} C_k^{L-1}}{2^L} = \frac{2^{L-1}}{2^L} = \frac{1}{2} \tag{4}$$

From the second line to the third line, we discard the term in which $k = 0$ as the term is 0 and does not affect the overall result. So the summation starts at $k = 1$ in the beginning of the third line.
The calculation above stands since all those f that can generate $\mathcal{D}$ is equally likely in probability, and we can see the choice of $\mathcal{A}$ is not relevant. So we conclude that $\mathbb{E}_f\Big\{E_{OTS}\big(\mathcal{A}(\mathcal{D}),f\big)\Big\} = \frac{1}{2} = \text{constant}$, regardless of $\mathcal{A}$.

Problem 4

We enumerate the combinations which produce five green 1's.

$$\begin{aligned}\mathbb{P}(\text{five green 1's}) =& \mathbb{P}(\text{five A's}) + \mathbb{P}(\text{four A's and one D's}) + \mathbb{P}(\text{three A's and two D's})\\ &+\mathbb{P}(\text{two A's and three D's}) + \mathbb{P}(\text{one A's and four D's}) + \mathbb{P}(\text{five D's})\\ =& \frac{C_5^5}{4^5} + \frac{C_4^5}{4^5} + \frac{C_3^5}{4^5} + \frac{C_2^5}{4^5} + \frac{C_1^5}{4^5} + \frac{C_0^5}{4^5}\\ =& \frac{32}{1024} = \frac{1}{32}\end{aligned}$$

or from simple observation, each 1 has probability $\frac{1}{2}$ of being green, so we have:

$$(\frac{1}{2})^5 = \frac{1}{32}$$

Problem 5

We start by some observations:

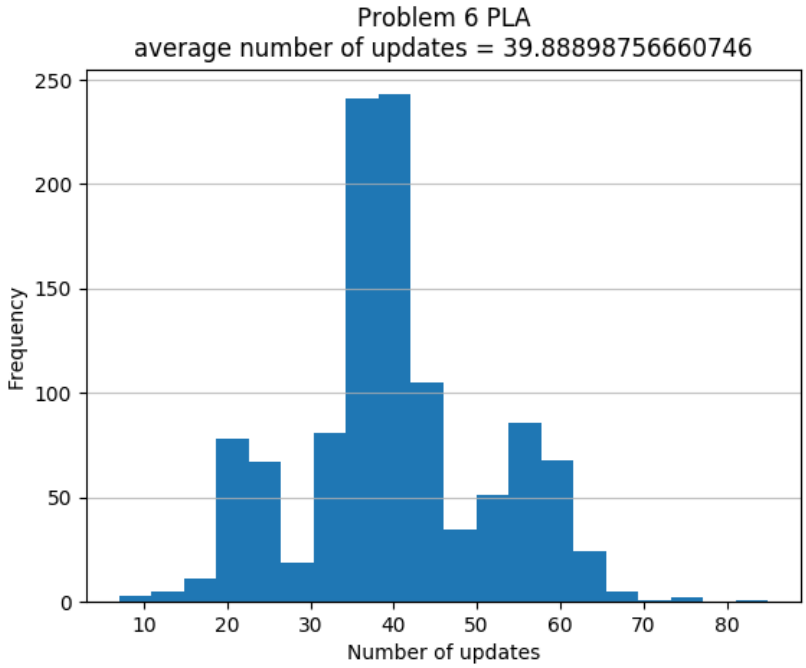
1. When 1 is purely green, so is 3, and vice versa. $\longrightarrow$ we can view 1 and 3 as one number.
2. When 4 is purely green, so is 6, and vice versa. $\longrightarrow$ we can view 4 and 6 as one number.
3. 1 and 4 can't be purely green at the same time.
4. 2 and 5 can't be purely green at the same time.

We now enumerate all possible situations when some number is purely green, and we have:

$$\begin{aligned}\mathbb{P}(\text{some number is purely green}) =& \mathbb{P}(1 \text{ is purely green}) + \mathbb{P}(2 \text{ is purely green}) + \mathbb{P}(4 \text{ is purely green}) + \mathbb{P}(5 \text{ is purely green}) \\ & - \mathbb{P}(1 \text{ and } 2 \text{ are purely green}) - \mathbb{P}(1 \text{ and } 5 \text{ are purely green}) \\ & - \mathbb{P}(2 \text{ and } 4 \text{ are purely green}) - \mathbb{P}(4 \text{ and } 5 \text{ are purely green}) \\ =& \frac{32}{1024} + \frac{32}{1024} + \frac{32}{1024} + \frac{32}{1024} - \frac{1}{1024} - \frac{1}{1024} - \frac{1}{1024} - \frac{1}{1024} \\ =& \frac{124}{1024} = \frac{31}{256}\end{aligned}$$

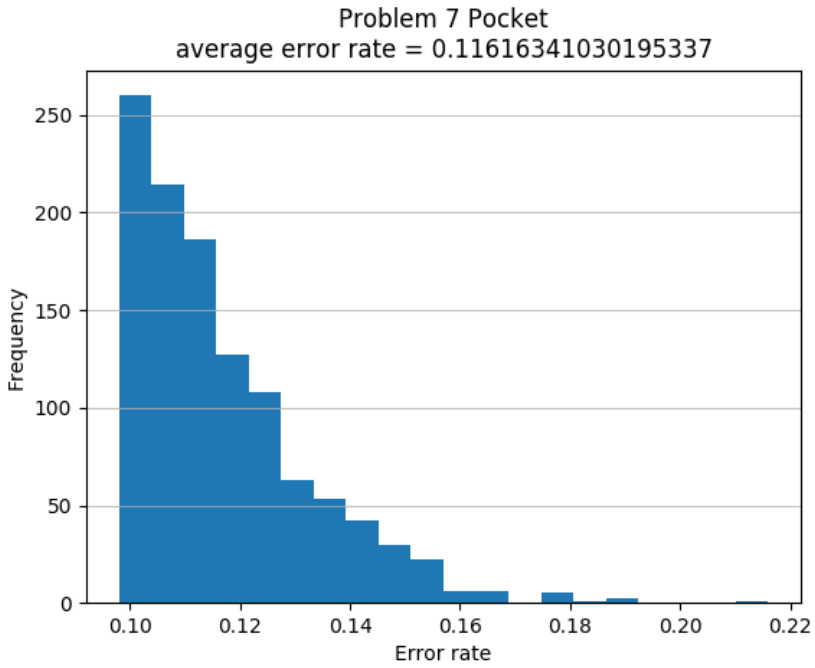
Although the previous problem asks for the probability of number 1 being purely green, the answer stands for all numbers from 1 to 6. Any number has probability $\frac{1}{32}$ of being purely green. And with simple calculation, any number has an expected value of 2.5 green's. So having 5 green's is what we refer to as "BAD DATA" in terms of machine learning, as E_{in} and E_{out} are far away. If we compare the situation to that in machine learning, with color green indicating correct answer to a sample, and numbers 1 to 6 represents $h_1, h_2, \ldots, h_6$ in the hypothesis set $\mathcal{H}$, then for every $h \in \mathcal{H}$, there is a probability of $\frac{1}{32}$ that it encounters "BAD DATA". But if look at all $h \in \mathcal{H}$ on a same set of data, then there's a probability of $\frac{31}{256}$, which is higher than $\frac{1}{32}$, that one of them ecounters "BAD DATA" and answers correctly to all samples. If we then conclude the h is our desired g, then we are mistaken as we know every $h \in \mathcal{H}$ is the same in terms of performance.

Problem 6



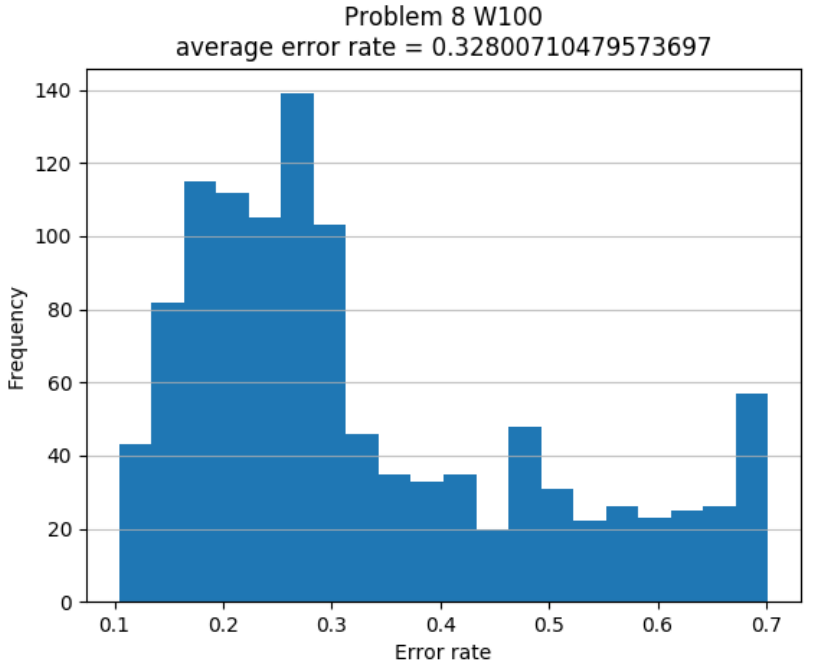
As shown in the figure above, average number of updates ≈ 39.889. Source code is named "P6_PLA.py" and is attached in the file uploaded to Ceiba. Usage is specified in Readme.txt in the same file.

Problem 7



As shown in the figure above, average error rate ≈ 0.116. Source code is named "P7_Pocket.py" and is attached in the file uploaded to Ceiba. Usage is specified in Readme.txt in the same file.

Problem 8



As shown in the figure above, average error rate ≈ 0.328. Source code is named "P8_W100.py" and is attached in the file uploaded to Ceiba. Usage is specified in Readme.txt in the same file.

Compared to the result in the previous problem, we note that the average error rate is higher and we have a more widespread error rate distribution. This is reasonable since in Pocket algorithm, we keep track of the best $\mathbf{w}$, improving it only when better one appears, while in this modified version, we are blindly updating $\mathbf{w}$, possibly getting a worse result. Even if the data is linearly separable and PLA guarantees convergence by updating $\mathbf{w}$ on errors, it does not guarantee every update decreases the overall errors made by $\mathbf{w}$, much less when data is not linearly separable. So when $\mathbf{w}_{100}$ is reached, it has worse performance on average and a more widespread error rate distribution.

Problem 9

No, the plan will not work. The proof is as follows:

First we introduce the notation:

- $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is the set of the original training data.
- $\mathcal{D}' = \{(x_1', y_1), (x_2', y_2), \ldots, (x_n', y_n)\}$ is the set of the modified training data, where $x_i' = \frac{1}{10}x_i$ for $i = 1, 2, \ldots, n$.
  - It's obvious that y_i for $i = 1, 2, \ldots, n$ does not change, so we take it for granted.
- $\mathbf{w}_i$ is the vector after the *i-th* update on $\mathcal{D}$, and $\mathbf{w}_0 = 0$, the zero vector, by the rule of PLA.
- $\mathbf{w}_i'$ is the vector after the *i-th* update on $\mathcal{D}'$, and $\mathbf{w}_0' = 0$, the zero vector, by the rule of PLA.

We will show that if PLA generates the sequence $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_f$ on $\mathcal{D}$, it will generate $\mathbf{w}_0', \mathbf{w}_1', \ldots, \mathbf{w}_f'$ on $\mathcal{D}'$, such that $\mathbf{w}_j' = \frac{1}{10}\mathbf{w}_j$ for $j = 1, 2, \ldots, f$.
Before the proof we point out a simple observation that multiplying a number by a positive constant does not change the sign of the number. So if $\mathbf{w}_j' = \frac{1}{10}\mathbf{w}_j$ for some j, then

$$y_i \cdot \text{sign}(\mathbf{w}_j \cdot x_i) = y_i \cdot \text{sign}(\tfrac{1}{10}\mathbf{w}_j \cdot \tfrac{1}{10}x_i) = y_i \cdot \text{sign}(\mathbf{w}_j' \cdot x_i') \text{ for } i = 1, 2, \ldots, n$$

With this idea in mind, we proceed to the proof and the proof is by induction on f:

Base case: $f = 0$

$$\mathbf{w}_0' = 0 = \tfrac{1}{10}\mathbf{w}_0$$

Inductive hypothesis: We assume for every $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_f$, there is $\mathbf{w}_0', \mathbf{w}_1', \ldots, \mathbf{w}_f'$, such that $\mathbf{w}_j' = \frac{1}{10}\mathbf{w}_j$ and thus $y_i \cdot \text{sign}(\mathbf{w}_j \cdot x_i) = y_i \cdot \text{sign}(\mathbf{w}_j' \cdot x_i')$ for $i = 1, 2, \ldots, n$ and $j = 0, 1, \ldots, f$.

Inductive step: Consider the case where $y_m \cdot \text{sign}(\mathbf{w}_f \cdot x_m) < 0$ for some $(x_m, y_m) \in \mathcal{D}$. The update rule of PLA gives

$$\mathbf{w}_{f+1} = \mathbf{w}_f + y_m \cdot x_m$$

By the inductive hypothesis, we also have $y_m \cdot \text{sign}(\mathbf{w}_f' \cdot x_m') < 0$, and the update rule gives

$$\mathbf{w}_{f+1}' = \mathbf{w}_f' + y_m \cdot x_m' = \tfrac{1}{10}\mathbf{w}_f + y_m \cdot \tfrac{1}{10}x_m = \tfrac{1}{10}(\mathbf{w}_f + y_m \cdot x_m) = \tfrac{1}{10}\mathbf{w}_{f+1}$$

The proof is complete and it shows that if PLA does not converge on $\mathcal{D}$, it does not converge on $\mathcal{D}'$ either; if it converges on $\mathcal{D}$, it also converges on $\mathcal{D}'$ after the same number of updates, so PLA does not converge faster.