Homework #1

RED CORRECTION: 10/08/2021 16:30

RELEASE DATE: 10/07/2021

DUE DATE: 10/21/2021, BEFORE 13:00 on Gradescope

QUESTIONS ARE WELCOMED ON THE NTU COOL FORUM.

You will use Gradescope to upload your choices and your scanned/printed solutions. For problems marked with (*), please follow the guidelines on the course website and upload your source code to Gradescope as well. You are encouraged to (but not required to) include a README to help the TAs check your source code. Any programming language/platform is allowed.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

This homework set comes with 16 problems and a total of 400 points. For each problem, there is one correct choice. If you choose the correct answer, you get 20 points; if you choose an incorrect answer, you get 0 points. For four of the secretly-selected problems, the TAs will grade your detailed solution in terms of the written explanations and/or code based on how logical/clear your solution is. Each of the four problems graded by the TAs counts as additional 20 points (in addition to the correct/incorrect choices you made). In general, each homework (except homework 0) is of a total of 400 points.

$20 \times 16 + 20 \times 4 = 400$

The Learning Problem

d

1. Which of the following problem is suited for machine learning if there is assumed to be enough associated data? Choose the correct answer; explain how you can possibly use machine learning to solve it.

[a] identifying the shortest distance path from Taipei to Kao-Hsiung optimization

[b] predicting the winning number of the super lotto probability ?

[c] calculating the total salary of all research assistants in a lab simple math

[d] sorting your e-commerce website users by their predicted chances of making a purchase in the next 7 days (may get a "g" from the purchasing data in the past.)

[e] none of the other choices

Modifications of PLA

2. One possible modification of PLA is to insert a per-iteration "learning rate" η_t to the update rule

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot \eta_t$$

There are many possibilities of setting the learning rate η_t. One is to consider how negative $y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)}$ is, and try to aggressively make $y_{n(t)}\mathbf{w}_{t+1}^T\mathbf{x}_{n(t)} > 0$; that is, let $\mathbf{w}_{t+1}$ correctly classify $(\mathbf{x}_{n(t)}, y_{n(t)})$. Another one is to conservatively decrease η_t so there is less oscillation during the final convergence stage. Which of the following update rules make $y_{n(t)}\mathbf{w}_{t+1}^T\mathbf{x}_{n(t)} > 0$? Choose the correct answer; explain your answer.

[a] $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot (2^{-t})$

[b] $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot 0.6211$

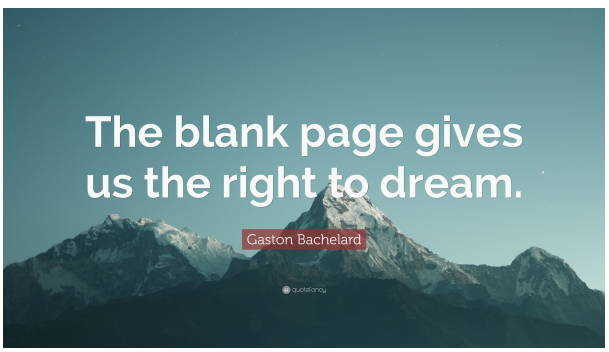
[c] $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot \left(\frac{-y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)}}{\|\mathbf{x}_{n(t)}\|^2}\right)$

[d] $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot \left(\frac{1}{1+t}\right)$

[e] $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)} \cdot \left\lfloor \frac{-y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)}}{\|\mathbf{x}_{n(t)}\|^2} + 1 \right\rfloor$

3. Dr. Separate decides to use one of the update rules in the previous problem for PLA. When the data set is linear separable, how many choices in the previous problem ensures halting with a "perfect line"? Choose the correct answer; explain the reason behind each perfect halting case.

[a] 1
[b] 2
[c] 3
[d] 4
[e] 5



(from quotefancy.com)

Extensions of PLA

4. Consider online spam detection with machine learning. We will represent each email $\mathbf{x}$ by the distinct words that it contains. In particular, assume that there are at most m distinct words in each email, and each word belongs to a big dictionary of size $d \geq m$. The i-th component x_i is defined as $[\![\text{word } i \text{ is in email } \mathbf{x}]\!]$ for $i = 1, 2, \ldots, d$, and $x_0 = 1$ as always. We will assume that d_+ of the words in the dictionary are more spam-like, and $d_- = d - d_+$ of the words are less spam-like. A simple function that classifies whether an email is a spam is to count $z_+(\mathbf{x})$, the number of more spam-like words with the email (ignoring duplicates), and $z_-(\mathbf{x})$, the number of less spam-like words in the email, and classify by

$$f(\mathbf{x}) = \text{sign}(z_+(\mathbf{x}) - z_-(\mathbf{x}) - 0.5).$$

That is, an email $\mathbf{x}$ is classified as a spam iff the integer $z_+(\mathbf{x})$ is more than the integer $z_-(\mathbf{x})$.

Assume that f can perfectly classify any email into spam/non-spam, but is unknown to us. We now run an online version of PLA to try to approximate f. That is, we maintain a weight vector $\mathbf{w}_t$ in the online PLA, initialized with $\mathbf{w}_0 = \mathbf{0}$. Then for every email $\mathbf{x}_t$ encountered at time t, the algorithm makes a prediction $\text{sign}(\mathbf{w}_t^T \mathbf{x}_t)$, and receives a true label y_t. If the prediction is not the same as the true label (i.e. a mistake), the algorithm updates $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t.$$

Otherwise the algorithm keeps $\mathbf{w}_t$ without updating

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t.$$

What is the maximum number of mistakes that the online PLA can make for this spam classification problem? Choose the tightest upper bound; explain your answer.

Note: For those who know the bag-of-words representation for documents, the representation we use is a simplification that ignores duplicates of the same word.

[a] $4(m+1)^2$

[b] $4(d+1)(m+1)$

[c] $(4d+1)(m+1)$

[d] $(4d+1)^2$

[e] $\frac{(d+1)m}{(d_+ - d_-)^2}$

5. For multiclass classification with K classes, the multiclass PLA maintains K weight vectors $\mathbf{w}_1^{(t)}$, $\mathbf{w}_2^{(t)}$, …, $\mathbf{w}_K^{(t)}$, where $^{(t)}$ is used to indicate the vectors in the t-th iteration. The goal is to obtain some hypothesis g represented by the solutions $\mathbf{w}_k^*$ returned from the algorithm, where g makes a prediction by

$$g(\mathbf{x}) = \text{argmax}_{k\in\{1,2,\dots,K\}} (\mathbf{w}_k^*)^T \mathbf{x} \ .$$

All weight vectors are initialized to $\mathbf{0}$ in the beginning. In iteration t, the multiclass PLA first finds a "mistake example" with index $n(t)$ that

$$y_{n(t)} \neq \text{argmax}_{k\in\{1,2,\dots,K\}} \left(\mathbf{w}_k^{(t)}\right)^T \mathbf{x}_{n(t)} \ .$$

Abbreviate $y_{n(t)}$ as y and let y' be the argmax result above. The multiclass PLA then updates weight vectors $\mathbf{w}_y^{(t)}$ and $\mathbf{w}_{y'}^{(t)}$ by

$$\begin{aligned} \mathbf{w}_y^{(t+1)} &\leftarrow \mathbf{w}_y^{(t)} + \mathbf{x}_{n(t)} \\ \mathbf{w}_{y'}^{(t+1)} &\leftarrow \mathbf{w}_{y'}^{(t)} - \mathbf{x}_{n(t)} \end{aligned}$$

If there are no more mistakes, return all $\mathbf{w}_k^{(t)}$ as the multiclass PLA solution $\mathbf{w}_k^*$.

Assume that $K = 2$. Given a linear separable data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where $y_n \in \{1, 2\}$. Run the binary PLA (initialized with the zero weight vector) taught in class on the data set with transformed labels $\tilde{y}_n = 2y_n - 3$ (so $\tilde{y}_n \in \{-1, +1\}$). Then, run the multiclass PLA above with the same order of examples (i.e. $n(t)$ across the two algorithms are the same). What is the relationship between $\mathbf{w}_{\text{PLA}}$ produced by the binary PLA, and the $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$ produced by the multiclass PLA? Choose the correct answer; explain your answer.

Note: If there is a tie in argmax, we will assume the "worst-case" scenario that argmax returns some k that is not equal to $y_{n(t)}$. Similarly, for the binary PLA, if sign(0) is encountered when checking the mistake on some $(\mathbf{x}_{n(t)}, y_{n(t)})$, we will assume the "worst-case" scenario that sign returns the sign that is not equal to $y_{n(t)}$.

[a] $\mathbf{w}_{\text{PLA}} = \mathbf{w}_1^* = -\mathbf{w}_2^*$
[b] $\mathbf{w}_{\text{PLA}} = -\mathbf{w}_1^* = \mathbf{w}_2^*$
[c] $\mathbf{w}_{\text{PLA}} = \mathbf{w}_2^* - \mathbf{w}_1^*$
[d] $\mathbf{w}_{\text{PLA}} = \mathbf{w}_1^* - \mathbf{w}_2^*$
[e] none of the other choices

The Learning Problems

6. Consider the following process: "We set up five cameras and capture video sequences from five different angles. The video sequences are timed but not specifically labeled. After collecting the sequences, we train a learning model that outputs a real-valued vector for each image in the video. The goal is that images that are taken at similar time stamps should be mapped to similar vectors, and images that are at distant time stamps should be mapped to different vectors. The learned model can then be combined with a variety of tasks, such as being mixed with some labeled data for object recognition, or some interactive environment that trains a robot to grab an object from the table." Which of the following best matches the process? Choose the best answer; explain your answer.

[a] active learning
[b] reinforcement learning
[c] multi-class classification
[d] self-supervised learning
[e] online learning

7. Consider formulating a learning problem for building a news tagger. First, we gather a training data set that consists of 1126 news articles, stored in utf8 encoding, and ask domain experts to tag each article by its categories. Each article can belong to several different categories. We also gather another 11265566 news articles from our database, without expert labeling. The learning algorithm is expected to learn from all the articles and tags (if any) to obtain a hypothesis that tags future news articles well. What learning problem best matches the description above? Choose the best answer; explain your answer.

[a] regression, unsupervised learning, active learning, concrete features

[b] text generation, semi-supervised learning, batch learning, concrete features

[c] multilabel classification, semi-supervised learning, batch learning, raw features

[d] regression, reinforcement learning, batch learning, concrete features

[e] multilabel classification, supervised learning, online learning, concrete features

(*We are definitely not hinting that you should build a news tagger this way. :-D*)

Off-Training-Set Error

8. As discussed in lecture 3, what we really care about is whether $g \approx f$ *outside* $\mathcal{D}$. For a set of "universe" examples $\mathcal{U}$ with $\mathcal{D} \subset \mathcal{U}$, the error *outside* $\mathcal{D}$ is typically called the Off-Training-Set (OTS) error

$$E_{\text{ots}}(h) = \frac{1}{|\mathcal{U} \setminus \mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{U}\setminus\mathcal{D}} [\![h(\mathbf{x}) \neq y]\!] .$$

Consider $\mathcal{U}$ with 6 examples

$\mathbf{x}$	y
$(-2, 2)$,	$+1$
$(0, 0)$,	$+1$
$(0, 3)$,	$+1$
$(1, 1)$,	-1
$(2, 1)$,	-1
$(3, 0)$,	-1

Run the process of choosing any three examples from $\mathcal{U}$ as $\mathcal{D}$, and learn a perceptron hypothesis (say, with PLA, or any of your "human learning" algorithm) to achieve $E_{\text{in}}(g) = 0$ on $\mathcal{D}$. Then, evaluate g outside $\mathcal{D}$. What is the smallest and largest $E_{\text{ots}}(g)$? Choose the correct answer; explain your answer.

[a] $(0, \frac{1}{3})$

[b] $(0, 1)$

[c] $(0, \frac{2}{3})$

[d] $(\frac{2}{3}, 1)$

[e] $(\frac{1}{3}, 1)$

Point Estimation

9. In lecture 3, we try to infer the unknown out-of-sample error $E_{\text{out}}(h)$ by the in-sample error $E_{\text{in}}(h)$ based on the observed data. This is an example of so-called *point estimation*. Formally, a *point estimator* is defined as a function of the data:

$$\hat{\theta} = g(x_1, \ldots, x_N),$$

where $\{x_1, \ldots, x_N\}$ is a set of N examples independent and identically (i.i.d.) generated from a distribution P. That is, $\hat{\theta}$ is also a random variable. Assume that the quantity that we try to infer is θ (determined by the distribution P), we attempt to use $\hat{\theta}$ as a "guess" of θ. For instance, $\theta = E_{\text{out}}(h)$ and $\hat{\theta} = E_{\text{in}}(h)$ in our lecture. One criteria to judge the goodness of $\hat{\theta}$ is whether it is *unbiased*. We call $\hat{\theta}$ unbiased iff

$$\mathbb{E}[\hat{\theta}] = \theta.$$

Consider the following point estimators. Which of them is *not* unbiased? Choose the correct answer; explain your answer.

[a] $\hat{\theta} = \frac{1}{N}\sum_{n=1}^{N} [\![h(\mathbf{x}_n) \neq y_n]\!]$ and $\theta = E_{\text{out}}(h) = \mathbb{E}_{\mathbf{x}\sim P}[\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$, where h is a fixed hypothesis, $f(\mathbf{x}) = y$ is the target function, and P is some fixed probability distribution that generates the data.

[b] $\hat{\theta} = \frac{1}{N}\sum_{n=1}^{N} x_n$. P is a Bernoulli distribution defined by $P(x) = \theta^x(1-\theta)^{(1-x)}$, where $x \in \{0, 1\}$.

[c] $\hat{\theta} = \max\{x_1, \ldots, x_N\}$. P is a discrete uniform distribution whose probability mass function is defined as $P(x) = \frac{1}{M}$ for $x \in \{1, \ldots, M\}$ $(M > N > 0)$. $\theta = M$.

[d] $\hat{\theta} = \frac{1}{N}\sum_{n=1}^{N} x_n^2$. P is a zero-mean *Gaussian distribution* and θ is its variance.

[e] none of the other choices (i.e. all of the other choices are unbiased)

Bad Data

10. Consider $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$, a target function $f(\mathbf{x}) = \text{sign}(x_1)$, a hypothesis $h_1(\mathbf{x}) = \text{sign}(2x_1 - x_2)$, and another hypothesis $h_2(\mathbf{x}) = \text{sign}(x_2)$. What is $(E_{\text{out}}(h_1), E_{\text{out}}(h_2))$ subject to the uniform distribution in $[-1, +1] \times [-1, +1]$ that generates $\mathbf{x}$? Choose the correct answer; explain your answer.

[a] $(\frac{1}{8}, \frac{1}{2})$
[b] $(\frac{7}{8}, \frac{1}{2})$
[c] $(\frac{1}{2}, \frac{1}{8})$
[d] $(\frac{1}{2}, \frac{7}{8})$
[e] $(\frac{1}{2}, \frac{1}{2})$

11. Following the previous problem, when drawing 4 examples independently and uniformly within $[-1, +1] \times [-1, +1]$ as $\mathcal{D}$, what is the probability that we get 4 examples such that $E_{\text{in}}(h_2) = E_{\text{in}}(h_1)$, including both the zero and non-zero E_{in} cases? Choose the correct answer; explain your answer.

Note: This is one of the BAD-data cases where we cannot distinguish the better-E_{out} hypothesis h_1 from the worse hypothesis h_2.

[a] $\frac{514}{4096}$
[b] $\frac{609}{4096}$
[c] $\frac{784}{4096}$
[d] $\frac{1126}{4096}$
[e] $\frac{1243}{4096}$

Multiple-Bin Sampling

12. We illustrate what happens with multiple-bin sampling with an experiment that use a dice (instead of a marble) to bind the six faces together. Please note that the dice is not meant to be thrown for random experiments. The probability below only refers to drawing the dices from the bag. Try to view each number as a *hypothesis*, and each dice as an *example* in our multiple-bin scenario. You can see that no single number is always green—that is, E_{out} of each hypothesis is always non-zero. In the following problem, we are essentially asking you to calculate the probability of the minimum $E_{\text{in}}(h_j) = 0$.

Consider four kinds of dice in a bag, with the same (super large) quantity for each kind.

- A: all even numbers are colored green, all odd numbers are colored orange
- B: $(1, 2, 6)$ are colored green, others are colored orange
- C: the number 6 is colored green, all other numbers are colored orange
- D: all primes are colored green, others are colored orange

If we draw 5 dices independently from the bag, what is the probability that we get *some number* that is purely green? Choose the correct answer; explain your answer.

[a] $\frac{512}{1024}$
[b] $\frac{454}{1024}$
[c] $\frac{333}{1024}$
[d] $\frac{243}{1024}$
[e] $\frac{32}{1024}$

four kinds of dice

Experiments with Perceptron Learning Algorithm

Next, we use an artificial data set to study PLA. The data set with $N = 100$ examples is in

`http://www.csie.ntu.edu.tw/~htlin/course/ml21fall/hw1/hw1_train.dat`

Each line of the data set contains one $(\mathbf{x}_n, y_n)$ with $\mathbf{x}_n \in \mathbb{R}^{10}$. The first 10 numbers of the line contains the components of $\mathbf{x}_n$ orderly, the last number is y_n. Please initialize your algorithm with $\mathbf{w} = \mathbf{0}$ and take sign(0) as -1.

13. (*) Please first follow page 29 of lecture 1, and add $x_0 = 1$ to every $\mathbf{x}_n$. Implement a version of PLA that randomly picks an example $(\mathbf{x}_n, y_n)$ in every iteration, and updates $\mathbf{w}_t$ if and only if $\mathbf{w}_t$ is incorrect on the example. Note that the random picking can be simply implemented *with replacement*—that is, the same example can be picked multiple times, even consecutively. Stop updating and return $\mathbf{w}_t$ as $\mathbf{w}_{\text{PLA}}$ if $\mathbf{w}_t$ is correct consecutively after checking $5N$ randomly-picked examples.

Hint: (1) The update procedure described above is equivalent to the procedure of gathering all the incorrect examples first and then randomly picking an example among the incorrect ones. But the description above is usually much easier to implement. (2) The stopping criterion above is a randomized, more efficient implementation of checking whether $\mathbf{w}_t$ makes no mistakes on the data set.

Repeat your experiment for 1000 times, each with a different random seed. What is the average squared length of $\mathbf{w}_{\text{PLA}}$? That is, $\|\mathbf{w}_{\text{PLA}}\|^2$? Choose the closest value.

[a] 360
[b] 390
[c] 420
[d] 450
[e] 480

391.88
386.95
386.13
395.25
387.32

include w_0 (?)

C **14.** (*) Scale up each $\mathbf{x}_n$ by 2, including scaling each x_0 from 1 to 2. Then, run PLA on the scaled examples for 1000 experiments. What is the average squared length of $\mathbf{w}_{\text{PLA}}$? That is, $\|\mathbf{w}_{\text{PLA}}\|^2$? Choose the closest value.

1574.32

1550.52

1570.56

[a] 1260
[b] 1410
[c] 1560
[d] 1710
[e] 1860

e **15.** (*) Scale down each $\mathbf{x}_n$ (including x_0) by $\|\mathbf{x}_n\|$, which makes each scaled example of length 1. Then, run PLA on the scaled examples for 1000 experiments. What is the average squared length of $\mathbf{w}_{\text{PLA}}$? That is, $\|\mathbf{w}_{\text{PLA}}\|^2$? Choose the closest value.

7.14

7.17

7.19

[a] 3.1
[b] 4.1
[c] 5.1
[d] 6.1
[e] 7.1

a **16.** (*) Set $x_0 = 0$ to every $\mathbf{x}_n$ instead of $x_0 = 1$, and do not do any scaling. This equivalently means not adding any x_0, and you will get a separating hyperplane that passes the origin. Repeat the 1000 experiments above. What is the average squared length of $\mathbf{w}_{\text{PLA}}$? That is, $\|\mathbf{w}_{\text{PLA}}\|^2$? Choose the closest value.

536.15

533.64

528.15

[a] 530
[b] 580
[c] 630
[d] 680
[e] 730