Homework #1

RELEASE DATE: 09/13/2023

DUE DATE: 09/27/2023, BEFORE 13:00 on GRADESCOPE

QUESTIONS ARE WELCOMED ON DISCORD (INFORMALLY) OR NTU COOL (FORMALLY).

You will use Gradescope to upload 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. 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.

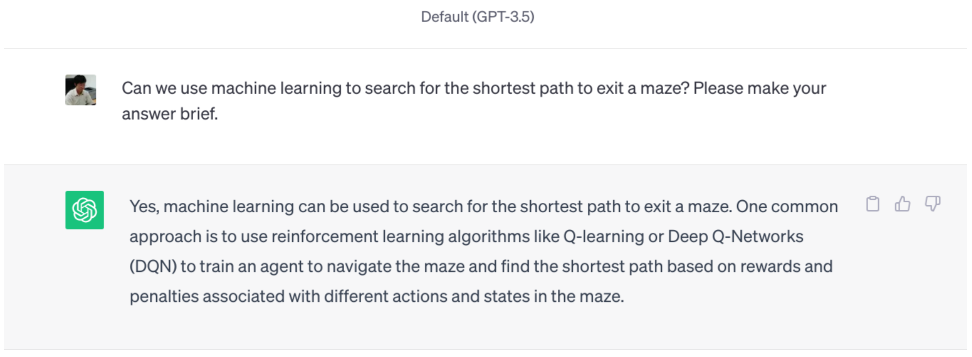
You should write your solutions in English or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

This homework set comes with 240 points and 20 bonus points. In general, every homework set would come with a full credit of 240 points, with some possible bonus points.

Learning Problems

1. (20 points) Describe an application of self-supervised learning with 10-20 English or Chinese sentences.

2. (20 points) The following answer was generated when asking chatGPT "*Can we use machine learning to search for the shortest path to exit a maze? Please make your answer brief.*" Do you agree or disagree with the answer? Provide your reasoning with 10-20 English or Chinese sentences.

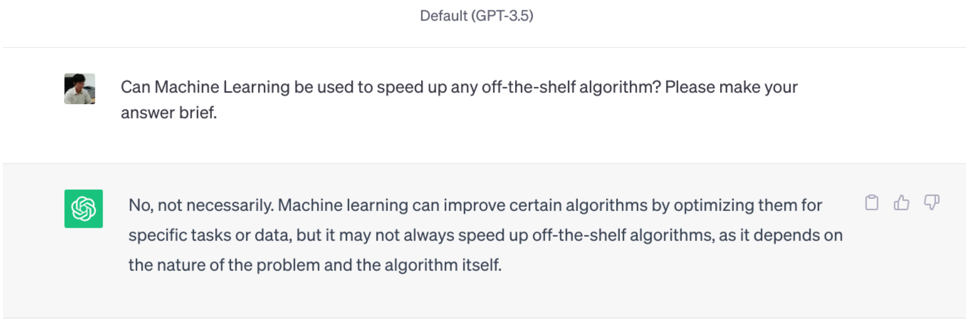


Default (GPT-3.5)

Can we use machine learning to search for the shortest path to exit a maze? Please make your answer brief.

Yes, machine learning can be used to search for the shortest path to exit a maze. One common approach is to use reinforcement learning algorithms like Q-learning or Deep Q-Networks (DQN) to train an agent to navigate the maze and find the shortest path based on rewards and penalties associated with different actions and states in the maze.

3. (20 points) The following answer was generated when asking chatGPT "*Can Machine Learning be used to speed up any off-the-shelf algorithm? Please make your answer brief.*" Somehow Google DeepMind announced that they discovered faster sorting algorithms with Machine Learning in June 2023 (`https://www.nature.com/articles/s41586-023-06004-9`). The discovery seems contradictory to the answer from chatGPT. Why is that? Provide your reasoning with 10-20 English or Chinese sentences.


Default (GPT-3.5)

Can Machine Learning be used to speed up any off-the-shelf algorithm? Please make your answer brief.

No, not necessarily. Machine learning can improve certain algorithms by optimizing them for specific tasks or data, but it may not always speed up off-the-shelf algorithms, as it depends on the nature of the problem and the algorithm itself.

Perceptron Learning Algorithm

4. (20 points) For the PLA algorithm introduced in class (page 8/22 of Lecture 2), assume that T_+ mistakes happened during $y_{n(t)} = 1$, and T_- mistakes happened during $y_{n(t)} = -1$. Express w_0, the zero-th component of the PLA solution, in terms of T_+ and T_-, and prove the result.

5. (20 points) 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 Perceptron Learning Algorithm (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

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

Prove or disprove that $(4d + 1)(m + 1)$ upper bounds the number of mistakes that the online PLA can make for this spam classification problem.

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.

6. (20 points) Before running PLA, our class convention adds $x_0 = 1$ to every $\mathbf{x}_n$ vector, forming $\mathbf{x}_n = (1, \mathbf{x}_n^{\text{orig}})$. Suppose that $x'_0 = -1$ is added instead to form $\mathbf{x}'_n = (-1, \mathbf{x}_n^{\text{orig}})$. Assume that running PLA on $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with a particular sequence $n(t), t = 1, 2, \ldots$ returns $\mathbf{w}_{\text{PLA}}$, and running PLA on on $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ with the same $n(t)$ returns $\mathbf{w}'_{\text{PLA}}$. Prove or disprove that $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$ are equivalent.

7. (20 points) Dr. Norman thinks PLA will be highly influenced by very long examples, as $\mathbf{w}_t$ changes drastically if $\|\mathbf{x}_{n(t)}\|$ is large. Hence, ze decides to preprocess the training data by normalizing each input vector i.e., $\mathbf{z}_n \leftarrow \frac{\mathbf{x}_n}{\|\mathbf{x}_n\|}$. What is PLA's upper bound on page 16/22 of Lecture 1 change with this preprocessing procedure in terms of $\rho_{\mathbf{z}} = \min_n \frac{y_n \mathbf{w}_f^T \mathbf{z}_n}{\|\mathbf{w}_f\|}$. Prove your answer.

8. (20 points) In PLA, we defined $\rho = \min\limits_n y_n \mathbf{w}_f^T \mathbf{x}_n$, and ρ is often called the unnormalized **margin** of $\mathbf{w}_f$. Margin is related to the minimal distance between $\mathbf{x}_n$ and hyperplane $\mathbf{w}_f$, and will be a main concept when we introduce the Support Vector Machine (SVM) later in this class. Before that, let us play with a variant of PLA, the Perceptron Algorithm using Margins (PAM). The difference between PAM and the original PLA is that PAM updates on $\mathbf{x}_n$ if

$$y_n \mathbf{w}_t^T \mathbf{x}_n \le \tau,$$

where $\tau \ge 0$ is the margin we would like to achieve.

For any given τ, assume that the data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is linearly τ-separable. That is, $\mathbf{w}_f$ satisfies $\rho > \tau$. Prove that PAM halts in a finite number of steps (*Hint: PLA is just a special case with* $\tau = 0$).

Experiments with Perceptron Learning Algorithm

Next, we use an artificial data set to study PLA. The data set with $N = 256$ examples is in

`http://www.csie.ntu.edu.tw/~htlin/course/ml23fall/hw1/hw1_train.dat`

Each line of the data set contains one $(\mathbf{x}_n, y_n)$ with $\mathbf{x}_n \in \mathbb{R}^{12}$. 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 $\text{sign}(0)$ as -1.

9. (20 points, *) Please first follow page 4/22 of Lecture 2, 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. Plot a histogram to visualize the distribution of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$. What is the median number of updates?

10. (20 points, *) Scale up each $\mathbf{x}_n$ by 11.26, including scaling each x_0 from 1 to 11.26. Then, run PLA on the scaled examples for 1000 experiments, each with a different random seed. Plot a histogram to visualize the distribution of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$. What is the median number of updates? Compare your result to that of Problem 9. Describe your findings.

11. (20 points, *) Set $x_0 = 11.26$ to every $\mathbf{x}_n$ instead of $x_0 = 1$, and do not do any scaling. Repeat the 1000 experiments above. Plot a histogram to visualize the distribution of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$. What is the median number of updates? Compare your result to that of Problem 9. Describe your findings.

12. (20 points, *) Set $x_0 = 1$ to every $\mathbf{x}_n$, and do not do any scaling. Modify your PLA above to a variant that keeps correcting the same example until it is perfectly classified. That is, when selecting an incorrect example $(\mathbf{x}_{n(t)}, y_{n(t)})$ for updating, the algorithm keeps using that example (that is, $n(t+1) = n(t)$) to update until the weight vector perfectly classifies the example (and each update counts!). Repeat the 1000 experiments above. Plot a histogram to visualize the distribution of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$. What is the median number of updates? Compare your result to that of Problem 9. Describe your findings.

Bonus: Does Normalization Help PLA?

13. (Bonus 20 points) In Problem 7, you showed that the upper bound of PLA changes when running on the normalized data instead of the original data. Does it mean that normalization can speed up PLA? Why or why not? Note that this is a bonus problem and TAs can set a high standard on your logical arguments when deciding whether to give you the points.