

Homework #1

RELEASE DATE: 09/13/2023

RED CORRECTION: 09/17/2023 17:00

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.

Ans :

Self-supervised learning 的應用比如設計 file system 的 prefetching 機制，prefetching 在 computer system 中十分常見，它將未來可能被訪問的數據預先讀取到 memory，以用來縮短 I/O access latency，並且這個問題很複雜，1. storage stack 很多層，不同層之間會互相干擾 2. I/O 指令帶來的訊息相較於其他指令(e.g., memory access) 較為粗糙，因此許多 researchers 透過自監督式學習，將某些 I/O access 隱藏起來以學習如上複雜的 I/O behavior，進而優化 prefetching 機制。

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.



Ans :

我認為我們可以透過機器學習來找到最短路徑。在過去，傳統的最短路徑演算法，如 Dijkstra's Algorithm 和 Floyd-Warshall Algorithm，常常需要大量的計算時間，Dijkstra's Algorithm 的時間複雜度是 $O((V + E) * \log(V))$ ，Floyd-Warshall Algorithm 的時間複雜度則是 $O(V^3)$ ，當圖很大時，這些演算法變得非常耗時。然而，透過機器學習，我們可以使用 **sampling** 等方法，無需完全探索圖的所有可能路徑，以大大縮短計算時間，另外也可以從歷史數據中學習，找到常見路徑和模式來減少重複搜索的時間。因此通過機器學習和樣本數據，我們可以優化尋找最短路徑的過程，解決目前最短路徑的所面臨的問題。

- 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.

Ans :

Google DeepMind 是在 2023 年 6 月宣布發現了使用機器學習加速排序算法的方法，而 ChatGPT 生成的答案基於 2021 年 9 月的訊息，在 google scholar 上，搜尋在 2000-2021 年間「intitle:"machine learning" and intitle:"sorting"」的論文僅 58 項結果，因此，這可能是 ChatGPT 回答錯的原因之一。另外，原因也可能為這個問題的問法不精確，以下為另一個問法下 ChatGPT 的回答，我們可以看出在另外一種問法下 ChatGPT 認為 Machine Learning 來加速 sorting 是可行的。



透過 machine learning 來加速 sorting 可以嗎



是的，可以使用機器學習來優化排序算法，以實現更快的排序速度。傳統的排序算法（例如快速排序、合併排序、插入排序等）通常具有較好的時間複雜度，但在某些情況下，可以使用機器學習來改進它們的性能，尤其是當排序數據具有特殊的分佈或結構時。

以下是一些方法和情境，其中機器學習可用於優化排序：

1. **自定義比較函數**：使用機器學習模型來訓練一個自定義的比較函數，該函數能夠根據數據的特點來決定排序順序，以減少比較的次數，從而提高排序效率。
2. **預測最佳排序算法**：使用機器學習模型來預測最適合當前數據分佈的排序算法，以根據情境自動選擇最佳算法。
3. **並行排序**：利用機器學習模型，可以優化數據的分佈，以便在多核或分佈式計算環境中實現並行排序，從而加速整個排序過程。
4. **硬體優化**：機器學習可以幫助優化特定硬體上的排序操作，例如 FPGA（現場可編程門陣列）或 GPU（圖形處理單元）等，以實現更快速的排序。

需要注意的是，這些方法的效果取決於數據的特點和排序的具體要求。傳統的排序算法通常在大多數情況下都很有效，但對於特殊情況，機器學習可能有助於進一步優化排序性能。

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.

Ans :

Handwritten derivation of the Perceptron Learning Algorithm solution for w_0 :

$$\begin{aligned}
 w_1 &= w_0 + x_{n(1)} y_{n(1)} \\
 &\vdots \\
 w_T &= w_{T-1} + x_{n(T)} y_{n(T)} \\
 &= \sum_{i=0}^{T-1} x_{n(i)} y_{n(i)} \\
 &= \sum_{i=0}^{T-1} \begin{bmatrix} 1 \\ x_{n(i),1} \\ \vdots \\ x_{n(i),d} \end{bmatrix} y_{n(i)} \\
 &= \sum_{i=0}^{T-1} \begin{bmatrix} y_{n(i)} \\ x_{n(i),1} y_{n(i)} \\ \vdots \\ x_{n(i),d} y_{n(i)} \end{bmatrix} = \begin{bmatrix} T_+ + T_- \\ x \\ \vdots \\ x \end{bmatrix}
 \end{aligned}$$

Ans : $T_+ + T_-$

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 [word i is in email \mathbf{x}] for $i = 1, 2, \dots, d$, 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.

$$\max_n \|X_n\| = \max_n \left\| \begin{bmatrix} x_{n,1} \\ \vdots \\ x_{n,d} \end{bmatrix} \right\|$$

∵ 一個文章最多 m 個獨特的字, 所以
 $x_{n,1} \dots x_{n,d}$ 至多 m 個 1

$$\therefore \left\| \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \right\|_{d+1} = \sqrt{m+1} \quad \text{--- ①}$$

$$\min_n y_n w_f^T X_n = \min_n \text{sign}(z_+(x) - z_-(x) - \frac{1}{2}) \cdot (z_+(x) - z_-(x) - \frac{1}{2})$$

as $z_+(x) = z_-(x)$ 有最小值 $\frac{1}{2}$ --- ②

$$\|W_f\|, \text{ 有一個 } W_f = \begin{bmatrix} \frac{1}{2} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}_{d+1}^{\text{bins}},$$

$$\begin{aligned} \text{sign}(W_f X_n) &= \text{sign}\left(\frac{1}{2} + \sum_{i=1}^{d_+} X_{n,i} - \sum_{i=1}^{d_-} X_{n,i}\right) \\ &= \text{sign}\left(z_+(X_n) - z_-(X_n) - \frac{1}{2}\right) \end{aligned}$$

= f(x) 可以完美分割資料

$$\|w_f\| = \sqrt{\frac{1}{4} + d_+ + d_-} = \sqrt{\frac{1}{4} + d} \quad (3)$$

$$1 \geq \frac{w_f^T w^T}{\|w_f\| \|w\|} \geq \frac{T \min_n y_n w_f^T x_n}{\|w_f\| \sqrt{T} \max_n \|x_n\|}$$

$$\geq \frac{\frac{1}{2} T \sqrt{T}}{\sqrt{\frac{1}{4} + d} \sqrt{T} \sqrt{m+1}}$$

$$\Rightarrow \sqrt{T} \leq 2 \sqrt{m+1} (\sqrt{\frac{1}{4} + d})$$

$$T \leq 4 \left(\frac{1}{4} + d \right) (m+1)$$

$$\leq (4d+1) (m+1) \quad \text{prove it}$$

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 $n(t), t = 1, 2, \dots$ with $\mathbf{w}_0 = \mathbf{0}$ returns \mathbf{w}_{PLA} , and running PLA on $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ with the same $n(t)$ with $\mathbf{w}_0 = \mathbf{0}$ returns \mathbf{w}'_{PLA} . Prove or disprove that \mathbf{w}_{PLA} and \mathbf{w}'_{PLA} are equivalent.

Ans :

Assume running PLA on $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ 的 最終 權重 為 $\mathbf{W} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_{\text{PLA}} \end{bmatrix}$; running PLA on $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ 的 最終 權重 為 $\mathbf{W}' = \begin{bmatrix} \mathbf{w}'_0 \\ \mathbf{w}'_{\text{PLA}} \end{bmatrix}$

使用數學歸納法證明 $\mathbf{w}_{\text{PLA}} = \mathbf{w}'_{\text{PLA}}$

當 $t=1$ 時, prove $\mathbf{w}_{\text{PLA}} = \mathbf{w}'_{\text{PLA}}$

$$\begin{aligned} \mathbf{w}_1 &= \mathbf{w}_0 + X_{n(1)} y_{n(1)} \\ &= \begin{bmatrix} 1 \\ x_{n(1)}^{\text{orig}} \end{bmatrix} y_{n(1)} \\ &= \begin{bmatrix} y_{n(1)} \\ y_{n(1)} x_{n(1)}^{\text{orig}} \end{bmatrix} = \begin{bmatrix} y_{n(1)} \\ \mathbf{w}'_{\text{PLA}} \end{bmatrix} \\ \mathbf{w}'_1 &= \mathbf{w}_0 + X'_{n(1)} y_{n(1)} \end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} 1 \\ x_{n(c)}^{\text{orig}} \end{bmatrix} y_{n(c)} \\
&= \begin{bmatrix} -y_{n(c)} \\ y_{n(c)} x_{n(c)}^{\text{orig}} \end{bmatrix} = \begin{bmatrix} -y_{n(c)} \\ W_{\text{PLA}}' \end{bmatrix} \\
W_{\text{PLA}}' &= y_{n(c)} x_{n(c)}^{\text{orig}} = W_{\text{PLA}}' \quad \text{得證} \\
\text{假設 } t=m \text{ 時, } W_{\text{PLA}}^m &= W_{\text{PLA}}^m \\
\frac{t}{b} t=m+1 \text{ 時, prove } W_{\text{PLA}}^{m+1} &= W_{\text{PLA}}^{m+1} \\
W_{m+1}' = \begin{bmatrix} x \\ W_{\text{PLA}}^{m+1} \end{bmatrix} &= W_m + x_{n(m+1)} y_{n(m+1)} \\
&= W_m + \begin{bmatrix} 1 \\ x_{n(m+1)}^{\text{orig}} \end{bmatrix} y_{n(m+1)} \\
&= \begin{bmatrix} y_{n(m+1)} + W_m^0 \\ x_{n(m+1)}^{\text{orig}} y_{n(m+1)} + W_{\text{PLA}}^m \end{bmatrix} \\
&= \begin{bmatrix} y_{n(m+1)} + W_m^0 \\ x_{n(m+1)}^{\text{orig}} y_{n(m+1)} + W_{\text{PLA}}^m \end{bmatrix} \\
&= \begin{bmatrix} B \\ W_{\text{PLA}}^{m+1} \end{bmatrix} \\
\text{B.L.E } W_{\text{PLA}}^{m+1} &= W_{\text{PLA}}^{m+1} \quad \text{得證} \\
\text{故可 prove } W_{\text{PLA}} &= W_{\text{PLA}}' \quad \text{X}
\end{aligned}$$

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 $\rho = \min_n \frac{y_n \mathbf{w}_f^T \mathbf{z}_n}{\|\mathbf{w}_f\|}$. Prove your answer.

Ans:

$$\begin{aligned}
 7. \quad \rho &= \min_n y_n \mathbf{w}_f^T \mathbf{z}_n; \quad R = \max_n \|\mathbf{z}_n\|^2 = 1 \\
 \mathbf{w}_f^T \mathbf{w}_1 &\geq \mathbf{w}_f^T \mathbf{w}_2 + \rho \\
 \mathbf{w}_f^T \mathbf{w}_2 &\geq \mathbf{w}_f^T \mathbf{w}_1 + \rho \Rightarrow \mathbf{w}_f^T \mathbf{w}_T \geq T \cdot \rho \\
 &\vdots \\
 \mathbf{w}_f^T \mathbf{w}_T &\geq \mathbf{w}_f^T \mathbf{w}_{T-1} + \rho \\
 \|\mathbf{w}_1\|^2 &\leq \|\mathbf{w}_0\|^2 + 1 \Rightarrow \|\mathbf{w}_T\|^2 \leq T \\
 &\vdots \\
 \|\mathbf{w}_T\|^2 &\leq \|\mathbf{w}_{T-1}\|^2 + 1 \\
 1 &\geq \frac{\mathbf{w}_f^T \mathbf{w}_T}{\|\mathbf{w}_f\| \|\mathbf{w}_T\|} \geq \frac{T \rho}{\|\mathbf{w}_f\| \sqrt{T}} \geq \sqrt{T} \rho \\
 \Rightarrow T &\leq \frac{1}{\rho^2} \quad \text{X}
 \end{aligned}$$

8. (20 points) In PLA, we defined $\rho = \min_n y_n \mathbf{w}_f^T \mathbf{x}_n$, 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

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

where $\tau \geq 0$ 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 $\tau = 0$*).

8.

$$y_{n(t)} w_f^T X_{n(t)} \geq \tau > 0$$

$$w_f^T w_{t+1} = w_f^T (w_t + y_{n(t)} X_{n(t)})$$

$$\geq w_f^T w_t + \tau$$

$$\geq w_f^T w_t$$

$$w_f^T w_1 \geq w_f^T w_0 + \tau$$

$$\vdots$$

$$w_f^T w_T \geq w_f^T w_{T-1} + \tau$$

$$\geq \frac{w_f^T w_T}{\|w_f\| \|w_T\|} \geq \frac{T \cdot \tau}{\sqrt{T} \max_n \|X_n\|}$$

$$\sqrt{T} \leq \frac{\max_n \|X_n\|}{\tau}$$

$$T \leq \left(\frac{\max_n \|X_n\|}{\tau} \right)^2$$

$$\leq \left(\frac{\max_n \|X_n\|}{\min_n y_n w_f^T X_n} \right)^2$$

prv it

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 12 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}_{PLA} if \mathbf{w}_t is correct consecutively after checking $5N$ randomly-picked examples.

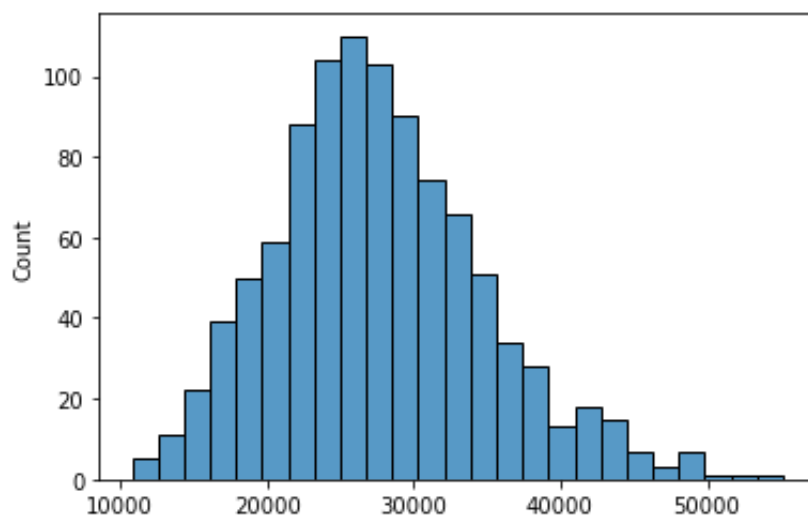
Hint: You can simply follow the algorithm above to solve this problem. But if you are interested in knowing why the algorithm above is somewhat equivalent to the PLA algorithm that you learned in class, here is some more information. (1) The update procedure described above is equivalent to the procedure 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. Passing $5N$ times of correctness checking means that \mathbf{w}_t is mistake-free with more than 99% of probability.

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}_{PLA} . What is the median number of updates?

Ans :

Median number of updates is 26923

Histogram :



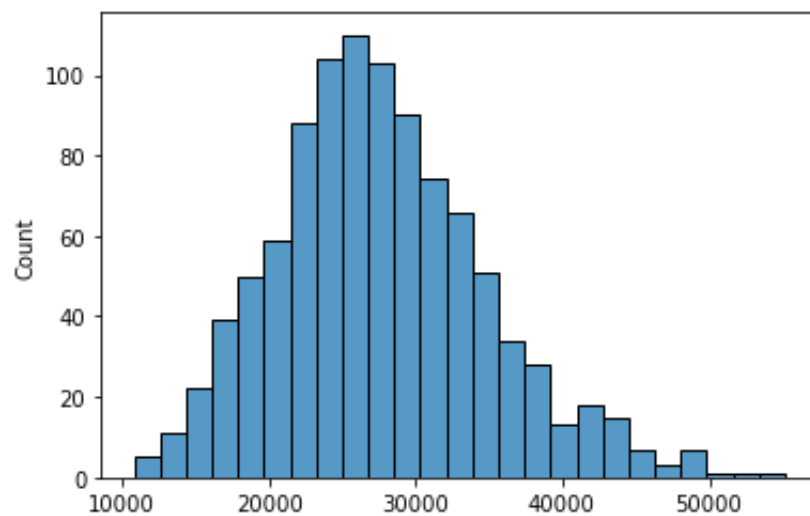
10. (20 points, *) Scale up each \mathbf{x}_n by 11.26, including scaling each x_0 from 1 to 11.26. Then, run PLA on the scaled examples for 1000 experiments, each with a different random seed. Plot a histogram to visualize the distribution of the number of updates needed before returning \mathbf{w}_{PLA}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

Ans :

Median number of updates is 26923 .

實驗可以看出 problem 9 與 problem 10 在每一個迭代 weight 相同，原因為 1. 兩個問題設定 random seed 的方法相同，因此兩個問題的輸入相同。2. scaling 只更改 weight 距離，不更改 classifier 的方向，因此不影響收斂次數。

Histogram :



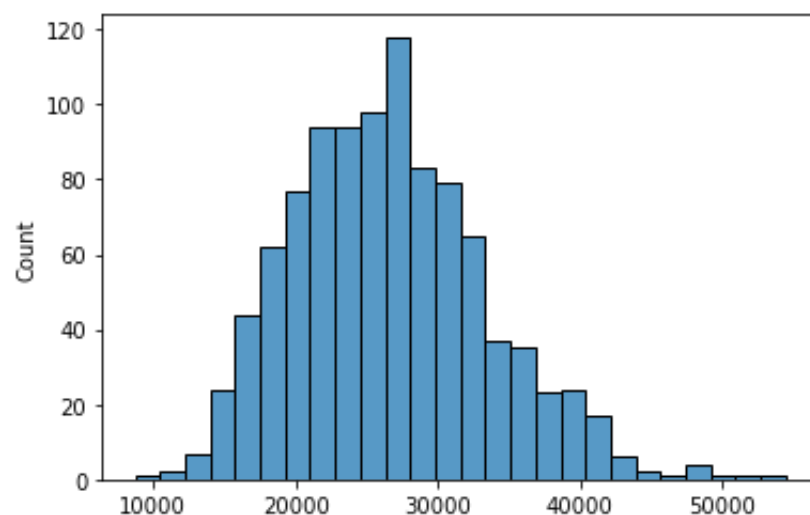
- 11.** (20 points, *) Set $x_0 = 11.26$ to every \mathbf{x}_n instead of $x_0 = 1$, and do not do any scaling. Repeat the 1000 experiments above. Plot a histogram to visualize the distribution of the number of updates needed before returning \mathbf{w}_{PLA} . What is the median number of updates? Compare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

Ans :

Median number of updates is 26243

與 problem 9 相比，可以看出當 bias 項會影響到迭代的次數，因為某些原本會被預測錯的點，因為 bias 改變反而預測對，此時 weight 的更新也隨之改變，而這項結果也正好對應到 problem 6 證明。

Histogram :



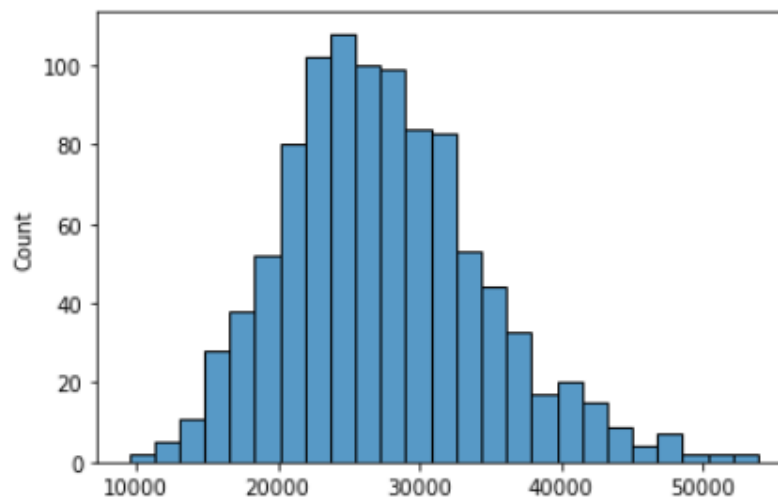
- 12.** (20 points, *) Set $x_0 = 1$ to every \mathbf{x}_n , and do not do any scaling. Modify your PLA above to a variant that keeps correcting the same example until it is perfectly classified. That is, when selecting an incorrect example $(\mathbf{x}_{n(t)}, y_{n(t)})$ for updating, the algorithm 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}_{PLA} . What is the median number of updates? Compare your result to that of Problem 9. Describe your findings.

Ans :

Median number of updates is 26809.5

與 problem 9 相比，可以看出迭代次數有所改變，但仔細看權重並無改變，原因為在 problem 12 必須保證完美分類，在原來停下時若沒有完美分類則會繼續執行，因此迭代次數有所改變。

Histogram :



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.