Machine Learning, Homework 1

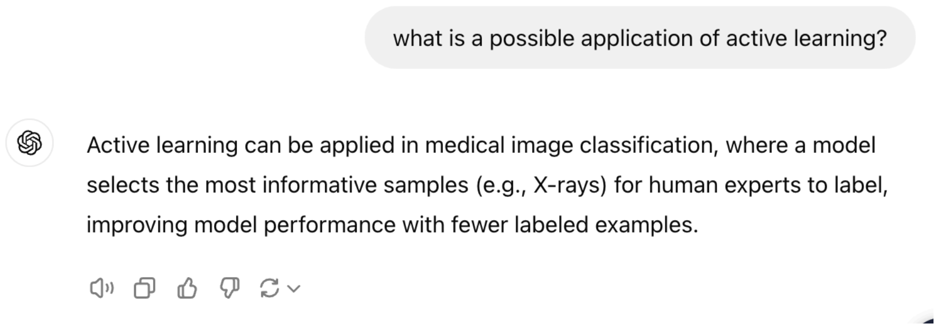
Lin, Jie (B11705048) 林杰

September 30, 2024

Problem 5

I agree with ChatGPT's answer. Active learning is a special case of machine learning in which the algorithm can ask questions. These questions can help the machine get labels that are considered useful. Since humans have more domain knowledge than machines, this is a good way for machines to learn, with fewer labels needed. ChatGPT mentioned that active learning can be applied in medical image classification. This is true since labeling many medical images is costly and time-consuming. Therefore, it's hard to get a bunch of labeled data for machines to learn. If the machine can ask the question strategically and obtain the important labels it needs, this can save many doctor's labeling time.

There is another advantage of active learning. By focusing on the most informative data points, machines don't need to rely on large, fully labeled datasets, which can be difficult or expensive to obtain, especially in fields like medicine. Additionally, the efficiency of the model's calculations may increase, as the algorithm learns from the most challenging or ambiguous cases.



what is a possible application of active learning?

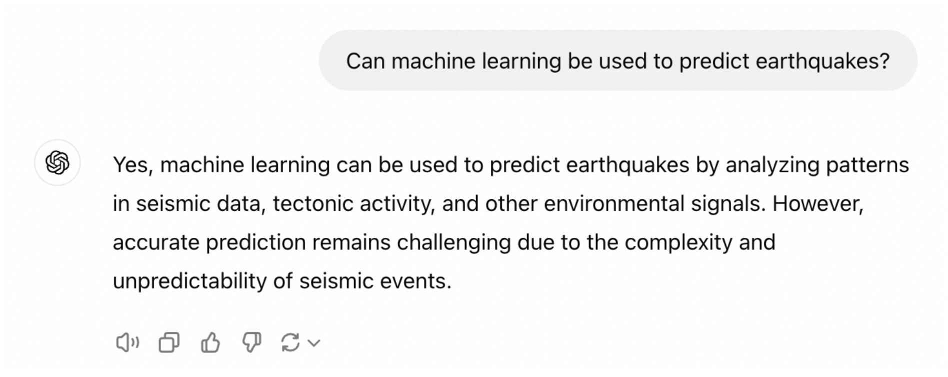
Active learning can be applied in medical image classification, where a model selects the most informative samples (e.g., X-rays) for human experts to label, improving model performance with fewer labeled examples.

Figure 1: The answer from ChatGPT

Problem 6

I agree with ChatGPT's answer. I will start by verifying the criterion taught in class about whether we should use machine learning. For now, there are no easy programmable definitions for the earthquake prediction problem. However, there may exist some underlying patterns that need to be learned. These patterns may be hidden in geographical data or environmental signals. ChatGPT mentioned that seismic data, tectonic activity, and other environmental signals may also be helpful. The patterns in the data may be so complicated that humans can not discover them, this is where machine learning is useful.

However, as ChatGPT mentioned, this task is still challenging. It requires some domain knowledge for experts to decide which data is useful for earth prediction. Furthermore, whether the pattern related to earthquakes will change over time or not remains unknown. Overall, I think ChatGPT provides a quite adequate answer.



Can machine learning be used to predict earthquakes?

Yes, machine learning can be used to predict earthquakes by analyzing patterns in seismic data, tectonic activity, and other environmental signals. However, accurate prediction remains challenging due to the complexity and unpredictability of seismic events.

Figure 2: The answer from ChatGPT

7. (20 points, human-graded) 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 $x'_0 = 2$ is added instead to form $\mathbf{x}'_n = (2, \mathbf{x}_n^{\text{orig}})$. Consider running PLA on $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ in a cyclic manner with the naïve cycle. That is, the algorithm keeps finding the next mistake in the order of $1, 2, \ldots, n, 1, 2, \ldots$. Assume that such a PLA with $\mathbf{w}_0 = \mathbf{0}$ returns $\mathbf{w}_{\text{PLA}}$, and running PLA on $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ with the same cyclic manner with $\mathbf{w}_0 = \mathbf{0}$ returns $\mathbf{w}'_{\text{PLA}}$. Prove or disprove that $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$ are equivalent. We define two weight vectors to be equivalent if they return the same binary classification output on every possible example.

Want to prove: $\vec{w}_t$ and $\vec{w}_t'$ can give the same classification result for any t.

In this problem, I set sign(0) to 1.

PLA base case, $t=0$.

At $t=0$, before any updates have been made:

- The initial weight vectors are both $\vec{w}_0 = \vec{0}$ for the original PLA and $\vec{w}_0' = \vec{0}$ for the modified PLA.
- Therefore, the classification results are identical since $\vec{w}_0 \cdot \vec{x}_{n(0)} = 0$ and $\vec{w}_0' \cdot \vec{x}_{n(0)} = 0$.

So at $t=0$, both settings can provide the same classification result. $\Rightarrow$ $\vec{w}_0$ and $\vec{w}_0'$ are equivalent.

When $t=1$:

Assume $\vec{w}_0$ will update based on the mistake $((1, \vec{x}_{n(0)}^{\,\text{orig}}), y_{n(0)})$:

$$\begin{bmatrix} w_{1,0} \\ w_{1,1} \\ \vdots \\ w_{1,d} \end{bmatrix} = \begin{bmatrix} w_{0,0} \\ w_{0,1} \\ \vdots \\ w_{0,d} \end{bmatrix} + y_{n(0)} \begin{bmatrix} 1 \\ x_{n(0),1} \\ \vdots \\ x_{n(0),d} \end{bmatrix} \Rightarrow \vec{w}_1 = \begin{bmatrix} y_{n(0)} \\ y_{n(0)} x_{n(0),1} \\ \vdots \\ y_{n(0)} x_{n(0),d} \end{bmatrix}$$

Assume $\vec{w}_0'$ will update based on the mistake $((2, \vec{x}_{n(0)}^{\,\text{orig}}), y_{n(0)})$:

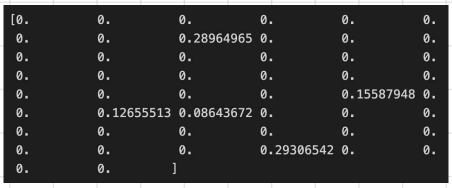
$$\begin{bmatrix} w'_{1,0} \\ w'_{1,1} \\ \vdots \\ w'_{1,d} \end{bmatrix} = \begin{bmatrix} w'_{0,0} \\ w'_{0,1} \\ \vdots \\ w'_{0,d} \end{bmatrix} + y_{n(0)} \begin{bmatrix} 2 \\ x_{n(0),1} \\ \vdots \\ x_{n(0),d} \end{bmatrix} \Rightarrow \vec{w}_1' = \begin{bmatrix} 2y_{n(0)} \\ y_{n(0)} x_{n(0),1} \\ \vdots \\ y_{n(0)} x_{n(0),d} \end{bmatrix}$$

For a random $\vec{x}_i$, $\text{sign}(\vec{w}_1'^{T} \vec{x}_i) = \text{sign}(\vec{w}_1^{T} \vec{x}_i)$ can not be guaranteed to be true, since the threshold of $\vec{w}_1'$ is double of the threshold of $\vec{w}_1$.
Therefore, the two PLAs can give different weight results, and may updates on different training data.

To conclude, we disprove that $\vec{w}_{PLA}$ and $\vec{w}_{PLA}'$ are equivalent.

I performed an experiment using the data given in problem 10.

$\vec{w}_{PLA}$ for $x_0 = 1$ (First 50 elements):

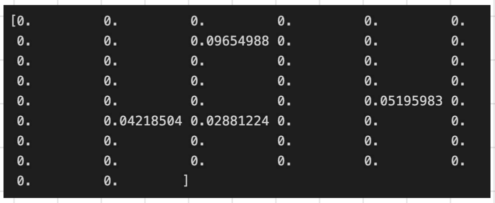


```
[0.         0.         0.         0.         0.         0.
 0.         0.         0.28964965 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.15587948 0.
 0.         0.12655513 0.08643672 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.29306542 0.         0.
 0.         0.        ]
```

$\xleftrightarrow{\text{different !!}}$

This serve as a counterexample.

$\vec{w}_{PLA}$ for $x_0 = 1$ (First 50 elements):



```
[0.         0.         0.         0.         0.         0.
 0.         0.         0.09654988 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.05195983 0.
 0.         0.04218504 0.02881224 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.        ]
```

8. (20 points, human-graded) 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 we scale every $\mathbf{x}_n$ by 3, and $x'_0 = 3$ is added instead to form $\mathbf{x}'_n = (3, 3\mathbf{x}_n^{\text{orig}})$. Consider running PLA on $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ in a cyclic manner with the naïve cycle. That is, the algorithm keeps finding the next mistake in the order of $1, 2, \ldots, n, 1, 2, \ldots$. Assume that such a PLA with $\mathbf{w}_0 = \mathbf{0}$ returns $\mathbf{w}_{\text{PLA}}$, and running PLA on $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ with the same cyclic manner with $\mathbf{w}_0 = \mathbf{0}$ returns $\mathbf{w}'_{\text{PLA}}$. Prove or disprove that $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$ are equivalent.

Want to prove:

1. $\vec{W}_t' = 3 \cdot \vec{W}_t$, for any t.
2. $\vec{w}_t'$ and $\vec{w}_t$ can always provide same classification result, for any t.

In this problem, I set $\text{sign}(0)$ to be 1.

PLA base case $t=0$:

At $t=0$, before any updates have been made:

- The initial weight vectors are both $\vec{W}_0 = \vec{0}$ for the original PLA and $\vec{w}_0' = \vec{0}$ for the modified PLA.
- Therefore, the classification results are identical since $\vec{w}_0 \cdot \vec{x}_{n(0)} = 0$ and $\vec{w}_0' \cdot \vec{x}_{n(0)}' = 0$.
- $\vec{W}_0' = 3 \cdot \vec{W}_0'$ holds.

So at $t=0$, both settings can provide the same classification result. $\Rightarrow$ $\vec{W}_0$ and $\vec{W}_0'$ are equivalent.

When $t=k$,

Assume that $\vec{W}_k$ and $\vec{W}_k'$ can provide the same classification result on every possible example, and also assume that $\vec{W}_k' = 3\vec{W}_k$.

When $t=k+1$,

Assume $\vec{W}_k$ will update based on the mistake $((1, \vec{x}_{n(k)}^{\text{orig}}), y_{n(k)})$

$$\begin{bmatrix} W_{k+1,0} \\ W_{k+1,1} \\ \vdots \\ W_{k+1,d} \end{bmatrix} = \begin{bmatrix} W_{k,0} \\ W_{k,1} \\ \vdots \\ W_{k,d} \end{bmatrix} + y_{n(k)} \begin{bmatrix} 1 \\ x_{n(k),1} \\ \vdots \\ x_{n(k),d} \end{bmatrix} \qquad \vec{W}_{k+1} = \vec{W}_k + y_{n(k)} \vec{x}_{n(k)}$$

Assume $\vec{W}_k'$ will update based on the mistake $((3, 3\vec{x}_{n(k)}^{\text{orig}}), y_{n(k)})$

$$\begin{bmatrix} W'_{k+1,0} \\ W'_{k+1,1} \\ \vdots \\ W'_{k+1,d} \end{bmatrix} = \begin{bmatrix} W'_{k,0} \\ W'_{k,1} \\ \vdots \\ W'_{k,d} \end{bmatrix} + y_{n(k)} \begin{bmatrix} 3 \\ 3x_{n(k),1} \\ \vdots \\ 3x_{n(k),d} \end{bmatrix} \qquad \vec{W}_{k+1}' = \vec{W}_k' + y_{n(k)} \cdot 3\vec{x}_{n(k)}$$

$$\vec{W}_{k+1}' = \vec{W}_k' + y_{n(k)} 3\vec{x}_{n(k)} = 3\vec{W}_k + y_{n(k)} 3\vec{x}_{n(k)} = 3(\vec{W}_k + y_{n(k)}\vec{x}_{n(k)}) = 3\vec{W}_{k+1}.$$

For a random $\vec{x}_i$, $\text{sign}(\vec{W}_{k+1}'^T \vec{x}_i) = \text{sign}(3 \cdot \vec{W}_{k+1}^T \vec{x}_i) = \text{sign}(\vec{W}_{k+1}^T \vec{x}_i)$

$\Rightarrow$ $\vec{W}_{k+1}$ and $\vec{W}_{k+1}'$ give the same result.

By mathematical induction, we can prove that:

1. Both hypotheses give the same classification result in any iteration t.
2. Both hypotheses give the same classification result after each update.

$\Rightarrow$ $\underline{\vec{W}_{PLA} \text{ and } \vec{W}_{PLA}' \text{ are equivalent.}}$ #

9. (20 points, human-graded) Consider online hatred article detection with machine learning. We will represent each article $\mathbf{x}$ by the distinct words that it contains. In particular, assume that there are at most m distinct words in each article, and each word belongs to a big dictionary of size $d \geq m$. The i-th component x_i is defined as $[\![\text{word } i \text{ is in article } \mathbf{x}]\!]$ 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 hatred-like, and $d_- = d - d_+$ of the words are less hatred-like. A simple function that classifies whether an artile is a hatred is to count $z_+(\mathbf{x})$, the number of more hatred-like words with the article (ignoring duplicates), and $z_-(\mathbf{x})$, the number of less hatred-like words in the article, and classify by

$$f(\mathbf{x}) = \text{sign}(z_+(\mathbf{x}) - z_-(\mathbf{x}) - 3.5).$$

That is, an article $\mathbf{x}$ is classified as a hatred iff the integer $z_+(\mathbf{x})$ is more than the integer $z_-(\mathbf{x})$ by 4.

Assume that f can perfectly classify any article into hatred/non-hatred, 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 article $\mathbf{x}_t$ 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

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

Derive an upper bound on the maximum number of mistakes that the online PLA can make for this hatred article classification problem. The tightness of your upper bound will be taken into account during grading.

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.

From the slide, $T \leq \left(\frac{R^2}{e^2}\right)$

$R^2 = \max_n \|\vec{x}_n\|^2 = m+1 \quad \ldots \text{(i)}$

$e = \min_n y_n \frac{\vec{w}_f^T}{\|\vec{w}_f\|} \vec{x}_n$, $e^2 = \min_n \left(\frac{\vec{w}_f^T \vec{x}_n}{\|\vec{w}_f\|}\right)^2 = \min_n \left(\frac{z_+(\vec{x}) - z_-(\vec{x}) - 3.5}{\|\vec{w}_f\|}\right)^2 = \frac{(0.5)^2}{\|\vec{w}_f\|^2}$

$z_+(\vec{x}) - z_-(\vec{x}) - 3.5$ is just a vector $(\vec{w}_f)$ with the 0-th element equals to 3.5. Each other elements equal to "+1" or "-1".

Therefore, $\|\vec{w}_f\| = (3.5)^2 + 1^2 \cdot d = d + (3.5)^2$

$\frac{1}{e^2} = \frac{\|\vec{w}_f\|^2}{0.5^2} = \frac{d + (3.5)^2}{(0.5)^2} \quad \ldots \text{(ii)}$

By (i) and (ii), $T \leq \left(\frac{R}{e}\right)^2 \leq \boxed{\frac{d + (3.5)^2}{(0.5)^2}(m+1)}$ #

$\hookrightarrow$ upper bound

Problem 10

From the histogram, we can observe that the distribution is right-skewed. Fewer updates are below 80, but a wide range of updates are observed between 80 and 115, with a sharp increase from 80 to 100. The median number of updates is 102.

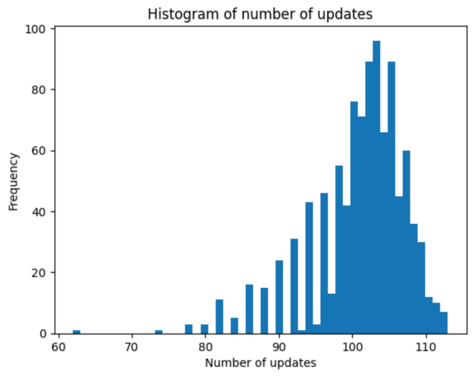
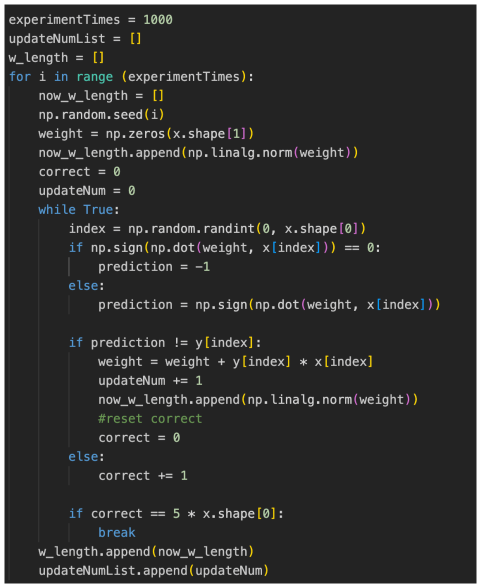


Figure 3: Histogram of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$



```
experimentTimes = 1000
updateNumList = []
w_length = []
for i in range (experimentTimes):
    now_w_length = []
    np.random.seed(i)
    weight = np.zeros(x.shape[1])
    now_w_length.append(np.linalg.norm(weight))
    correct = 0
    updateNum = 0
    while True:
        index = np.random.randint(0, x.shape[0])
        if np.sign(np.dot(weight, x[index])) == 0:
            prediction = -1
        else:
            prediction = np.sign(np.dot(weight, x[index]))

        if prediction != y[index]:
            weight = weight + y[index] * x[index]
            updateNum += 1
            now_w_length.append(np.linalg.norm(weight))
            #reset correct
            correct = 0
        else:
            correct += 1

        if correct == 5 * x.shape[0]:
            break
    w_length.append(now_w_length)
    updateNumList.append(updateNum)
```

Figure 4: Code for problem 10

Problem 11

From the functions I plotted, we can observe that the increment rate for the functions is decreasing. This means that when the number of updates is larger, the length of the weight increases slower when updating.

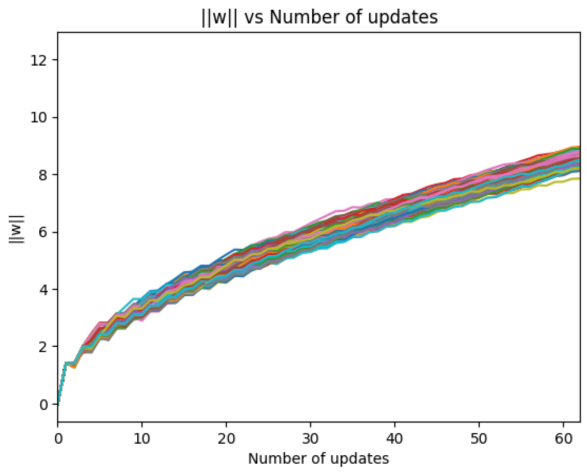
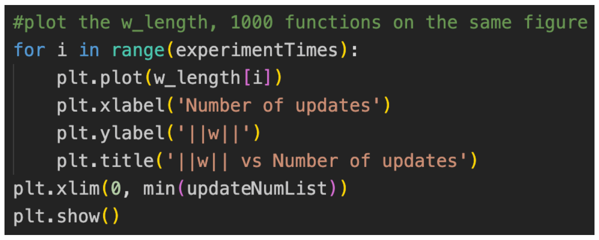


Figure 5: $||\mathbf{w}_t||$ vs number of updates



```
#plot the w_length, 1000 functions on the same figure
for i in range(experimentTimes):
    plt.plot(w_length[i])
    plt.xlabel('Number of updates')
    plt.ylabel('||w||')
    plt.title('||w|| vs Number of updates')
plt.xlim(0, min(updateNumList))
plt.show()
```

Figure 6: Code for problem 11

Problem 12

The histogram shows that the distribution is right-skewed, identical to the one we observed in problem 10. Fewer updates are below 80, but a wide range of updates are observed between 80 and 115, with a sharp increase from 80 to 100. The median number of updates is still 102, the same as the median in problem 10. Therefore, changing the correction rule does not influence the number of updates.

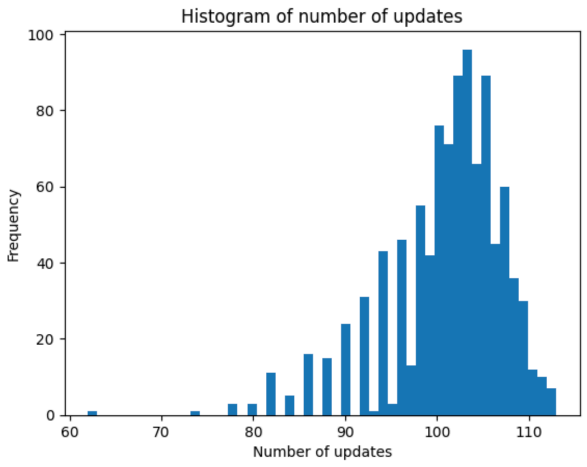
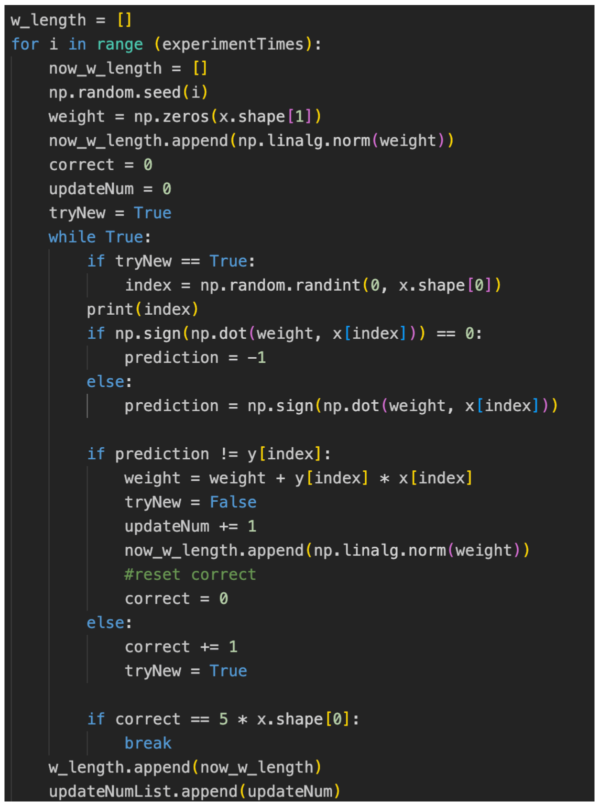


Figure 7: Histogram of the number of updates needed before returning $\mathbf{w}_{\text{PLA}}$



```python
w_length = []
for i in range (experimentTimes):
    now_w_length = []
    np.random.seed(i)
    weight = np.zeros(x.shape[1])
    now_w_length.append(np.linalg.norm(weight))
    correct = 0
    updateNum = 0
    tryNew = True
    while True:
        if tryNew == True:
            index = np.random.randint(0, x.shape[0])
        print(index)
        if np.sign(np.dot(weight, x[index])) == 0:
            prediction = -1
        else:
            prediction = np.sign(np.dot(weight, x[index]))

        if prediction != y[index]:
            weight = weight + y[index] * x[index]
            tryNew = False
            updateNum += 1
            now_w_length.append(np.linalg.norm(weight))
            #reset correct
            correct = 0
        else:
            correct += 1
            tryNew = True

        if correct == 5 * x.shape[0]:
            break
    w_length.append(now_w_length)
    updateNumList.append(updateNum)
```

Figure 8: Code for problem 12

13. (Bonus 20 points, human-graded) When PLA makes an update on a misclassified example $(\mathbf{x}_{n(t)}, y_{n(t)})$, the new weight vector $\mathbf{w}_{t+1}$ does not always classify $(\mathbf{x}_{n(t)}, y_{n(t)})$ correctly. Consider a variant of PLA that makes an update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \frac{1}{10} y_{n(t)} \mathbf{x}_{n(t)} \cdot \left\lfloor \frac{-10 y_{n(t)} \mathbf{w}_t^T \mathbf{x}_{n(t)}}{\|\mathbf{x}_{n(t)}\|^2} + 1 \right\rfloor .$$

First, prove that $\mathbf{w}_{t+1}$ always correctly classifies $(\mathbf{x}_{n(t)}, y_{n(t)})$ after the update. Second, prove that such a PLA halts with a perfect hyperplane if the data is linearly separable. (*Hint: Check Problem 12.*)

Note that:

$y_i(\vec{w}^T \vec{x}_i) > 0 \iff \vec{x}_i$ is classified correctly.

$$y_{n(t)}(\vec{w}_{t+1}^T \vec{x}_{n(t)}) = y_{n(t)}\left[\left(\vec{w}_t + \frac{1}{10} y_{n(t)} \vec{x}_{n(t)} \cdot \left\lfloor \frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor \right)^T \vec{x}_{n(t)}\right]$$

$$= y_{n(t)}(\vec{w}_t^T \vec{x}_{n(t)}) + y_{n(t)}^2 \left(\frac{1}{10} \vec{x}_{n(t)} \cdot \left\lfloor \frac{\overbrace{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}^{\text{positive}}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor \right)^T \vec{x}_{n(t)}$$

$$> y_{n(t)}(\vec{w}_t^T \vec{x}_{n(t)}) + \left(\frac{1}{10} \vec{x}_{n(t)} \left(\frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2}\right)\right)^T \vec{x}_{n(t)}$$

$$= y_{n(t)}(\vec{w}_t^T \vec{x}_{n(t)}) + \left(\vec{x}_{n(t)} \frac{-y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2}\right)^T \vec{x}_{n(t)} = y_{n(t)}(\vec{w}_t^T \vec{x}_{n(t)}) - y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)} = 0$$

Since $y_{n(t)} \vec{w}_{t+1}^T \vec{x}_{n(t)} > 0$, we can say that $\vec{w}_{t+1}$ always correctly classifies $(\vec{x}_{n(t)}, y_{n(t)})$ after the update.

Next, prove that the PLA halts with a perfect hyperplane if the data is linearly seperable.

Assume $\vec{w}_f$ is the perfect hyperplane, so for every $\vec{x}_n$, $y_{n(t)} \vec{w}_f^T \vec{x}_{n(t)} \geq \min_n y_n \vec{w}_f^T \vec{x}_n > 0$.

$$\vec{w}_f^T \vec{w}_{t+1} = \vec{w}_f^T \left(\vec{w}_t + \frac{1}{10} y_{n(t)} \vec{x}_{n(t)} \cdot \left\lfloor \frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor \right)$$

$$\geq \vec{w}_f^T \vec{w}_t + \min_n \frac{1}{10} y_n \vec{w}_f^T \vec{x}_n \left\lfloor \frac{-10 y_n \vec{w}_t^T \vec{x}_n}{\|\vec{x}_n\|^2} + 1 \right\rfloor$$

$$\|\vec{w}_{t+1}\|^2 = \left\| \vec{w}_t + \frac{1}{10} y_{n(t)} \vec{x}_{n(t)} \left\lfloor \frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor \right\|^2$$

$$= \|\vec{w}_t\|^2 + \frac{2}{10} y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)} \left\lfloor \frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor + \left\| \frac{1}{10} y_{n(t)} \vec{x}_{n(t)} \left\lfloor \frac{-10 y_{n(t)} \vec{w}_t^T \vec{x}_{n(t)}}{\|\vec{x}_{n(t)}\|^2} + 1 \right\rfloor \right\|^2$$

$$\leq \|\vec{w}_t\|^2 + \max_n \left\| \frac{1}{10} y_n \vec{x}_n \left\lfloor \frac{-10 y_n \vec{w}_t^T \vec{x}_n}{\|\vec{x}_n\|^2} + 1 \right\rfloor \right\|^2$$

Take $\min_n \frac{1}{10} y_n \vec{w}_f^T \vec{x}_n \left\lfloor \frac{-10 y_n \vec{w}_t^T \vec{x}_n}{\|\vec{x}_n\|^2} + 1 \right\rfloor = e\|\vec{w}_f\|$

$\max_n \left\| \frac{1}{10} y_n \vec{x}_n \left\lfloor \frac{-10 y_n \vec{w}_t^T \vec{x}_n}{\|\vec{x}_n\|^2} + 1 \right\rfloor \right\|^2 = R^2$

By the slide, after T mistake corrections, $T \leq \frac{R^2}{e^2}$

#