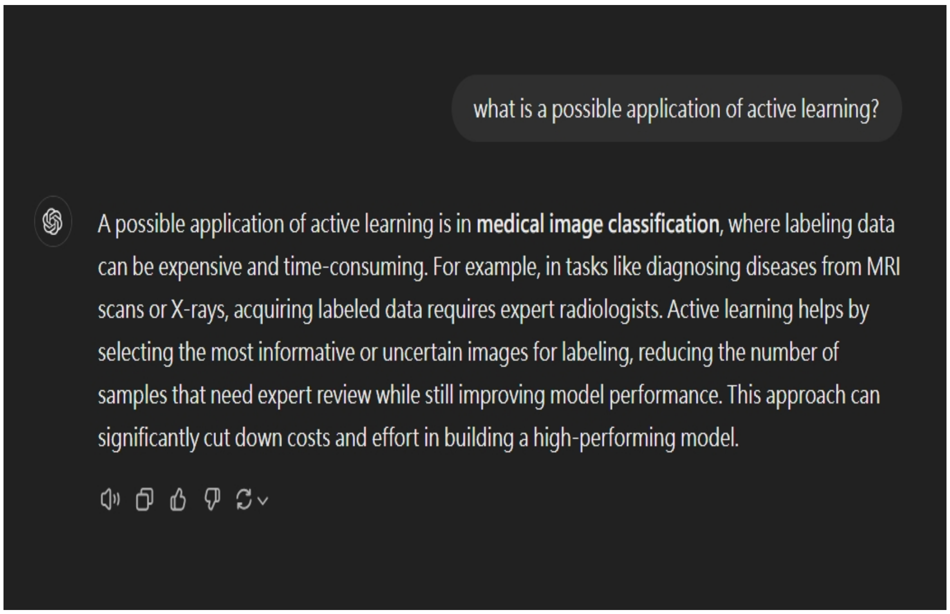
ML HW1

R13922146 葉富銘

Problem 5



With previous knowledge, my understanding of active learning is it actively uses query strategy to select data points and acquire the label that can improve the performance the most, hence it is useful when labeling data is hard, costly and time consuming. According to its answer, medical image classification, labeling data is expensive and time-consuming, which sounds reasonable. Takes MRI scans for example, labeling the scan requires a radiologist to carefully examine and annotate the images. It is highly technical and needs a lot of experience, which take a lot of cost. So it is suitable to use active learning, such as using uncertainty sampling to select the most uncertain scans which the model is unsure whether a tumor is present, hence a few labeling can get lots of information and improve significantly. Therefore, I agree with what it says.

Problem 6

can machine learning be used to predict earthquakes

Yes, machine learning (ML) is increasingly being explored to predict earthquakes, though it remains a highly challenging task due to the complexity and chaotic nature of seismic activity. Here are some ways ML is being applied:

1. **Pattern Recognition in Seismic Data:** ML algorithms can analyze historical seismic data to identify patterns that precede earthquakes. These patterns might include changes in the frequency of small tremors or other geological signals. Algorithms like deep learning models are particularly useful for identifying subtle patterns in large datasets.
2. **Sensor Data Analysis:** Seismic sensors collect vast amounts of data, including information about ground motion, strain, and vibrations. ML models can process these signals in real-time to detect anomalies that may indicate an upcoming earthquake. Techniques like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are used for this purpose.
3. **Geophysical Feature Correlation:** Machine learning models can be trained to correlate various geophysical features (e.g., tectonic movements, rock compositions, fault lines) with earthquake occurrences. Researchers use algorithms like Random Forests and Support Vector Machines (SVMs) to predict area ↓ higher risk of earthquakes based on these features.
4. **Aftershock Prediction:** ML models have shown promise in predicting aftershocks, which follow large earthquakes. For example, Gradient Boosting algorithms and Neural Networks can be trained on aftershock patterns from past events to estimate the likelihood of future aftershocks in a specific region.

Challenges

- **Complexity of Earthquake Mechanics:** Earthquake prediction is difficult because many factors are involved, and the processes governing earthquake initiation are still not fully understood.
- **Lack of Predictable Signals:** Unlike weather, where long-term patterns can be observed and modeled, earthquakes may not provide consistent or easily detectable precursors.
- **Data Scarcity:** While there is data on past earthquakes, it's still limited compared to the needs of machine learning models, especially for large, rare seismic events.

Despite these challenges, machine learning holds potential to improve earthquake risk assessments, early warning systems, and better understand seismic activities over time.

Its answer matchs the essence of machine learning learned in class : it exists some ”underlying pattern” precede earthquake to be learned; it has no programmable definition since the cause of earthquake is much more complicated, there are lots of factor result in earthquake; there is data about the pattern, according to its saying, ground motion,strain and vibrations. So it is suitable to use machine learning. Although there might have some challenges like the complexity of its mechanics, data scarcity, we can add extra technique to try to deal with it. For example, intergrate some domain knowledge such as tectonic plate movement with the models, or use data augmentation to get more data. In my opinion, since we have already known that it is so hard to predict earthquake, It is not surprising that it must encounter some challenge while using machine learning, we can take ML to be a potiential, powerful tool to help us further discover some unknown property, which can help us configure the mechanism better. Therefore, I agree with what it says.

Problem 7

To disprove that $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$ are equivalent, we only need to find an example where we can find two different hyperplanes according to $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$.

For simplicity, assume $\mathbf{x}_n \in \mathbf{R}^2$, assume $\text{sign}(0) = 1$. We only need to find two different lines according to $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$.

Construct an original data set with two data points $(\mathbf{x}_1^{\text{orig}}, y_1)$ and $(\mathbf{x}_2^{\text{orig}}, y_2)$,

$$\mathbf{x}_1^{\text{orig}} = [1, 2], \quad y_1 = -1$$
$$\mathbf{x}_2^{\text{orig}} = [5, 1], \quad y_2 = 1$$

1. **Running PLA on** $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{w}_0 = \mathbf{0}$:

   $\mathbf{x}_1 = [1, 1, 2]$, $\mathbf{x}_1 = [1, 5, 1]$

   Cycle 1 :
   $\mathbf{x}_1 : \text{sign}(\mathbf{w}_0^T \mathbf{x}_1) \neq y_1$, $\mathbf{w}_1 = \mathbf{w}_0 + y_1\mathbf{x}_1 = [-1, -1, -2]$
   $\mathbf{x}_2 : \text{sign}(\mathbf{w}_0^T \mathbf{x}_2) \neq y_2$, $\mathbf{w}_2 = \mathbf{w}_1 + y_2\mathbf{x}_2 = [0, 4, -1]$

   Cycle 2 :
   $\mathbf{x}_1 : \text{sign}(\mathbf{w}_2^T \mathbf{x}_1) \neq y_1$, $\mathbf{w}_3 = \mathbf{w}_2 + y_1\mathbf{x}_1 = [-1, 3, -3]$
   $\mathbf{x}_2 : \text{sign}(\mathbf{w}_3^T \mathbf{x}_2) = y_2$,

   Cycle 3 :
   $\mathbf{x}_1 : \text{sign}(\mathbf{w}_3^T \mathbf{x}_1) = y_1$
   $\mathbf{x}_2 : \text{sign}(\mathbf{w}_3^T \mathbf{x}_2) = y_2$

   Stop, return $\mathbf{w}_{\text{PLA}} = [-1, 3, -3]$.
   $\text{L1} : -1 + 3x_1 - 3x_2 = 0$

2. **Running PLA on** $\{(\mathbf{x}'_n, y_n)\}_{n=1}^N$ with $\mathbf{w}_0 = \mathbf{0}$:

   $\mathbf{x}'_1 = [2, 1, 2]$, $\mathbf{x}'_2 = [2, 5, 1]$

   Cycle 1 :
   $\mathbf{x}'_1 : \text{sign}(\mathbf{w}_0^T \mathbf{x}'_1) \neq y_1$, $\mathbf{w}_1 = \mathbf{w}_0 + y_1\mathbf{x}'_1 = [-2, -1, -2]$
   $\mathbf{x}'_2 : \text{sign}(\mathbf{w}_0^T \mathbf{x}'_2) \neq y_2$, $\mathbf{w}_2 = \mathbf{w}_1 + y_2\mathbf{x}'_2 = [0, 4, -1]$

   Cycle 2 :
   $\mathbf{x}'_1 : \text{sign}(\mathbf{w}_2^T \mathbf{x}'_1) \neq y_1$, $\mathbf{w}_3 = \mathbf{w}_2 + y_1\mathbf{x}'_1 = [-2, 3, -3]$
   $\mathbf{x}'_2 : \text{sign}(\mathbf{w}_3^T \mathbf{x}'_2) = y_2$,

   Cycle 3 :
   $\mathbf{x}'_1 : \text{sign}(\mathbf{w}_3^T \mathbf{x}'_1) = y_1$
   $\mathbf{x}'_2 : \text{sign}(\mathbf{w}_3^T \mathbf{x}'_2) = y_2$

   Stop, return $\mathbf{w}_{\text{PLA}} = [-2, 3, -3]$

   $\text{L1} : -2 + 3x_1 - 3x_2 = 0$

We find two lines according to $\mathbf{w}_{\text{PLA}}$ and $\mathbf{w}'_{\text{PLA}}$, and can easily find a points $\left(\frac{3}{2}, 1\right)$ which these lines classify it differently, hence proved.

Problem 8

Assume $\text{sign}(0) = 1$, $\mathbf{w}_0 = \mathbf{w}'_0 = \mathbf{0}$

1. $\text{sign}(\mathbf{w}_0^T\mathbf{x}_n) = \text{sign}(\mathbf{w}_0^T\mathbf{x}'_n) = 1$, if it updates $\mathbf{w}_1 = \mathbf{w}_0 + y_1\mathbf{x}_1$, it will also update $\mathbf{w}'_1 = \mathbf{w}'_0 + y_1\mathbf{x}'_1$ $= \mathbf{w}'_0 + y_1 3\mathbf{x}_1 = 3\mathbf{w}_1$.

2. We can find that $\mathbf{x}'_n$ is just the scalar multiple by 3 of $\mathbf{x}_n$, that is. $\mathbf{x}'_n = 3\mathbf{x}_n$

   We can prove two things:

   (a) For t ≥ 1, if $\mathbf{w}'_t = 3\mathbf{w}_t$, $\text{sign}(\mathbf{w}'^T_t\mathbf{x}'_n) = \text{sign}(3\mathbf{w}_t^T 3\mathbf{x}_n) = \text{sign}(\mathbf{w}_t^T\mathbf{x}_n)$

   (b) For t ≥ 1, if $\mathbf{w}'_t = 3\mathbf{w}_t$, when encounter an update, $\mathbf{w}_{t+1} = \mathbf{w}_t + y_n\mathbf{x}_n$, $\mathbf{w}'_{t+1} = \mathbf{w}'_t + y_n\mathbf{x}'_n$ $= 3\mathbf{w}_t + 3y_2\mathbf{x}_n = 3\mathbf{w}_{t+1}$.

According to the above, we can find that at any time, $\mathbf{w}'_t = 3\mathbf{w}_t$, and hence $\mathbf{w}'_{\text{PLA}} = 3\mathbf{w}_{\text{PLA}}$, and the hyperplane constructed from them are actually the same, so they are equivalent.

Problem 9

According to the PLA mistake bound proved in class, we can derive an upper bound $T \leq \left(\frac{R}{\rho}\right)^2$, T is the number of corrections.

Given the definition of R and ρ in lecture, we can rewrite the above inequality into $T \leq \left(\frac{\|\mathbf{w}_f\|^2 \max\limits_n \|\mathbf{x}_n\|^2}{\left(\min\limits_n \mathbf{w}_f^T\mathbf{x}_n\right)^2}\right)$

- For $\|\mathbf{w}_f\|^2$, $f(x) = \text{sign}(z_+(x) - z_-(x) - 3.5) = \text{sign}(\mathbf{w}_f^T x)$, wf needs to satisfy the following condition :

$$\mathbf{w}_i = \begin{cases} 1 & \text{for more hatred-like words } (d^+ \text{ of these}) \\ -1 & \text{for less hatred-like words } (d^- \text{ of these}) \end{cases} \tag{1}$$

$$\mathbf{w}_f^T\mathbf{0} = -3.5 \text{ (threshold)} \tag{2}$$

  We can then derive $\mathbf{w}_f$ to be a vector that contain one element = -3.5, d_+ 1s and d_- -1s, and hence $\|\mathbf{w}_f\|^2 = d_+ + d_- + 3.5^2 = d + 12.25$.

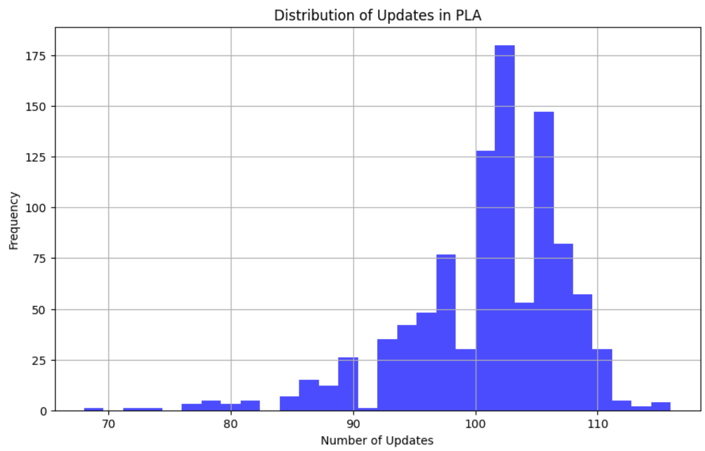
- $\max\limits_n \|\mathbf{x}_n\|^2$ = m + 1 (since at most m words can be 1 or -1, plus $x_0 = 1$)

- $\mathbf{w}_f^T\mathbf{x}_n$ will be the smallest when it is the closet positive or negative example, that is, $z_+(x) - z_-(x) - 3.5 = 0.5$ or -0.5. So $\left(\min\limits_n \mathbf{w}_f^T\mathbf{x}_n\right)^2 = 0.5^2 = 0.25$

Given above, we can derive an upperbound T $\leq$ 4(d + 12.25)(m + 1).

Problem 10

Most of the number of updates fall between 90 and 110. In fact, I have calculated the mean and the median of number of updates, which are both approximately around 100.



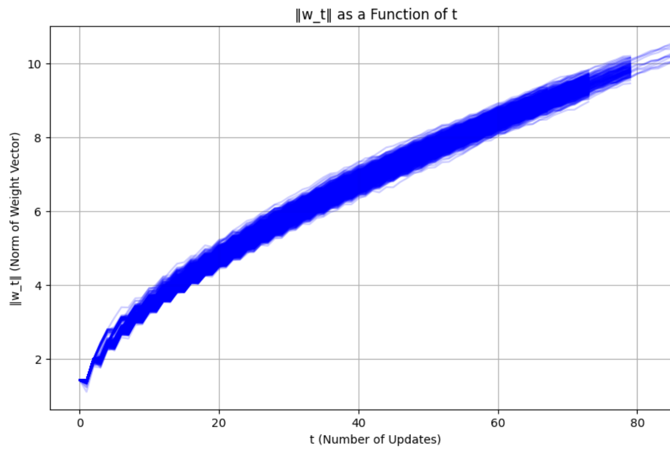
(a)

```python
def pla(data, labels):
    x_0 = torch.tensor([1], device=device)    # x[0] = 1
    w = torch.zeros(D + 1, device=device)    # w_0 = [0, 0, ... 0]

    # sign(0) = -1
    sign_0 = torch.tensor([0], device=device)
    sign_minus_1 = torch.tensor([-1], device=device)

    consecutive_correct = 0
    target_correct = 5 * N
    updates = 0

    while consecutive_correct < target_correct:
        # random sampling
        i = secrets.randbelow(N)
        x_n = torch.cat((x_0, data[i]))
        y_n = labels[i]

        product = torch.sign(torch.dot(w, x_n))
        product = sign_minus_1 if product == sign_0 else product

        if product != y_n:
            consecutive_correct = 0
            updates += 1
            w += y_n * x_n
        else:
            consecutive_correct += 1

        # if torch.sign(torch.dot(w, x_n)) != y_n:
        #     consecutive_correct = 0
        #     updates += 1
        #     w += y_n * x_n
        # else:
        #     consecutive_correct += 1

    return updates
```

(b)

Problem 11

As the number of updates increase, $\|\mathbf{w}_t\|$ also increase. I guess it's beacuse that w_0 is initialized to $\mathbf{0}$, as it keeps updating using different $\mathbf{x}_n$, more and more zero elelments in the vector are changed into non-zero, so $\|w_t\|$ increase. It also shows that T_{min} has decreased, but most of them are close to 80, indicate that the number of updates in each experiment are very close.



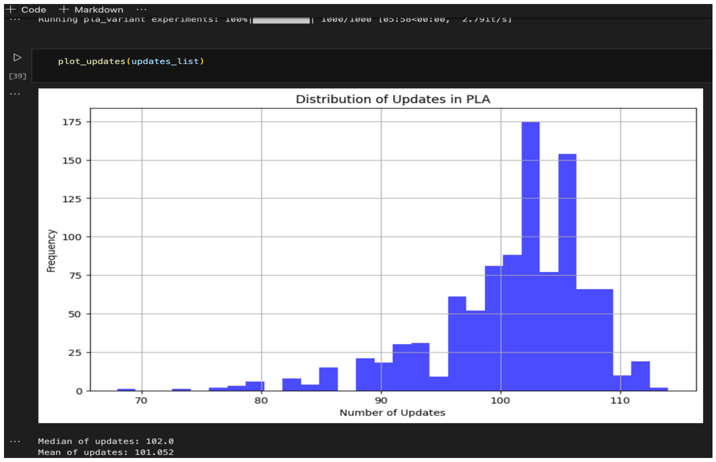
(c)

```python
def pla_norm(data, labels, T_min):
    x_0 = torch.tensor([1], device=device)   # x[0] = 1
    w = torch.zeros(D + 1, device=device)   # w_0 = [0, 0, ... 0]
    func_w_t_norm = []              # record ||w_t||

    # sign(0) = -1
    sign_0 = torch.tensor([0], device=device)
    sign_minus_1 = torch.tensor([-1], device=device)

    consecutive_correct = 0
    target_correct = 5 * N
    updates = 0

    # labels_tensor = torch.tensor(labels, dtype=torch.float64)

    while consecutive_correct < target_correct:
        # random sampling
        i = secrets.randbelow(N)
        x_n = torch.cat((x_0, data[i]))
        y_n = labels[i]

        product = torch.sign(torch.dot(w, x_n))
        product = sign_minus_1 if product == sign_0 else product

        if product != y_n:
            consecutive_correct = 0
            updates += 1
            w += y_n * x_n

            # record ||w_t|| if times of update <= T_min
            if updates <= T_min:
                func_w_t_norm.append(float(torch.norm(w)))
        else:
            consecutive_correct += 1

    T_min = min(T_min, updates)

    return updates, T_min, func_w_t_norm
```

(d)

Problem 12

The distribution of the number of updates seems similar to problem 10, so do its mean (101.052) and median (102), which indicate that maybe the updating mechanism has small effect. That is, most of the example can be corrected by only one update.



(e)

```python
def pla_variant(data, labels):
    x_0 = torch.tensor([1], device=device)      # x[0] = 1
    w = torch.zeros(D + 1, device=device)      # w_0 = [0, 0, ... 0]

    # sign(0) = -1
    sign_0 = torch.tensor([0], device=device)
    sign_minus_1 = torch.tensor([-1], device=device)

    consecutive_correct = 0
    target_correct = 5 * N
    updates = 0

    while consecutive_correct < target_correct:
        # random sampling
        i = secrets.randbelow(N)
        x_n = torch.cat((x_0, data[i]))
        y_n = labels[i]

        product = torch.sign(torch.dot(w, x_n))
        product = sign_minus_1 if product == sign_0 else product

        if product != y_n:
            # update w untill correct
            while product != y_n:
                updates += 1
                w += y_n * x_n
                product = torch.sign(torch.dot(w, x_n))
                product = sign_minus_1 if product == sign_0 else product

            # reset consecutive_correct
            consecutive_correct = 1
        else:
            consecutive_correct += 1

    return updates
```

(f)