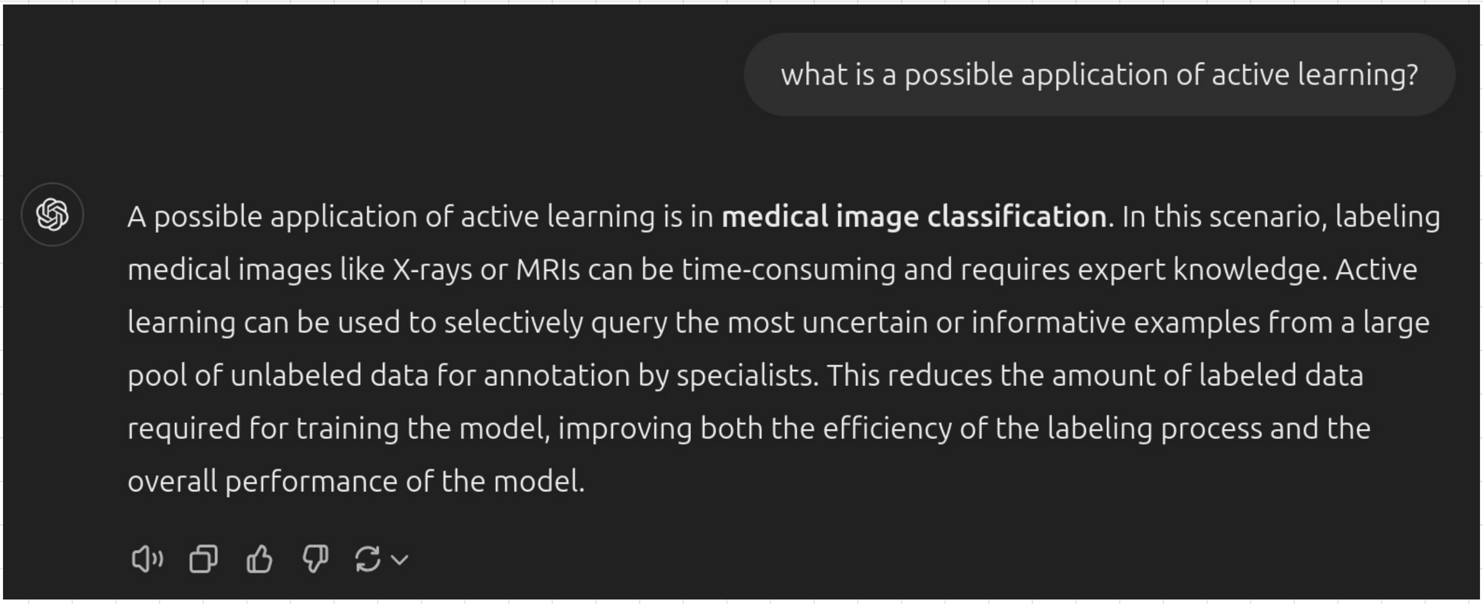
5. no collaborators

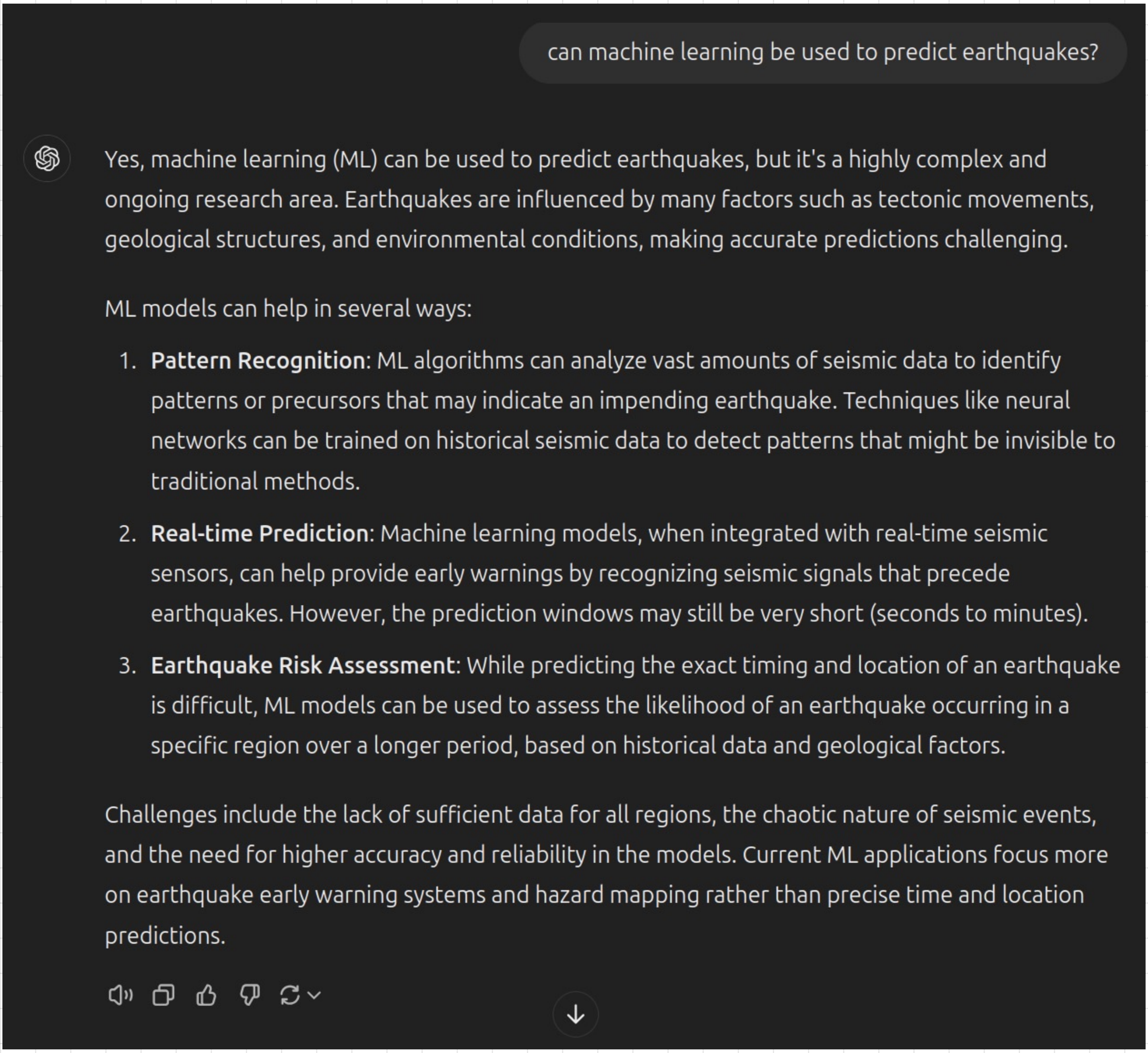


what is a possible application of active learning?

A possible application of active learning is in **medical image classification**. In this scenario, labeling medical images like X-rays or MRIs can be time-consuming and requires expert knowledge. Active learning can be used to selectively query the most uncertain or informative examples from a large pool of unlabeled data for annotation by specialists. This reduces the amount of labeled data required for training the model, improving both the efficiency of the labeling process and the overall performance of the model.

I agree with the agent. As mentioned by the agent, labeling medical images can be time-consuming. Thus, the supervised learning is not suitable for the medical image classification. On top of that, it also requires expert knowledge. Thus, active learning may perform better than unsupervised and semi-supervised learning. In conclusion, medical image classification is a possible application of active learning.

6. no collaborators



can machine learning be used to predict earthquakes?

Yes, machine learning (ML) can be used to predict earthquakes, but it's a highly complex and ongoing research area. Earthquakes are influenced by many factors such as tectonic movements, geological structures, and environmental conditions, making accurate predictions challenging.

ML models can help in several ways:

1. **Pattern Recognition**: ML algorithms can analyze vast amounts of seismic data to identify patterns or precursors that may indicate an impending earthquake. Techniques like neural networks can be trained on historical seismic data to detect patterns that might be invisible to traditional methods.
2. **Real-time Prediction**: Machine learning models, when integrated with real-time seismic sensors, can help provide early warnings by recognizing seismic signals that precede earthquakes. However, the prediction windows may still be very short (seconds to minutes).
3. **Earthquake Risk Assessment**: While predicting the exact timing and location of an earthquake is difficult, ML models can be used to assess the likelihood of an earthquake occurring in a specific region over a longer period, based on historical data and geological factors.

Challenges include the lack of sufficient data for all regions, the chaotic nature of seismic events, and the need for higher accuracy and reliability in the models. Current ML applications focus more on earthquake early warning systems and hazard mapping rather than precise time and location predictions.

I agree with the agent. According to Page 6 of Lecture 1, predicting earthquakes meets the following three conditions. First, existing some underlying pattern to be learned: tectonic movements, geological structures, and environmental conditions. Second, no programmable (easy) definition. Last, data: history of earthquake. Because of the lack of sufficient data for all regions, predicting earthquakes may be implemented for some regions only.

7.

setting $sign(0) = 1$

Collaborators:

B11901073 林禹融

Let $\underline{w}_{PLA}^T = (w_{PLA,0}, w_{PLA,1}, w_{PLA,2}, \dots, w_{PLA,d})$

$\underline{w}'^T_{PLA} = (w'_{PLA,0}, w'_{PLA,1}, w'_{PLA,2}, \dots, w'_{PLA,d})$

$h(\underline{x}_n) = sign(\underline{w}_{PLA}^T \underline{x}_n) = sign(\underline{w}_{PLA}^T (1, \underline{x}_n^{orig}))$

$h(\underline{x}'_n) = sign(\underline{w}'^T_{PLA} \underline{x}'_n) = sign(\underline{w}'^T_{PLA} (2, \underline{x}_n^{orig}))$

counter example:

$(w_{PLA,1}, w_{PLA,2}, \dots, w_{PLA,d}) \cdot \underline{x}_n^{orig} = -2$

$w_{PLA,i} = w'_{PLA,i} \quad \forall\, i \in \mathbb{Z} \wedge 0 \leq i \leq d$

$w_{PLA,0} = w'_{PLA,i} = 1$

$$\Rightarrow \begin{cases} h(\underline{x}_n) = sign(\underline{w}_{PLA}^T \underline{x}_n) = sign(1 + (-2)) = sign(-1) = -1 \\ h(\underline{x}'_n) = sign(\underline{w}'^T_{PLA} \underline{x}'_n) = sign(2 + (-2)) = sign(0) = 1 \end{cases}$$

$\therefore$ $\underline{w}_{PLA}$ and $\underline{w}'_{PLA}$ are not equivalent

Q.E.D.

8.

no collaborators

setting $\text{sign}(0) = 0$

Let $\underline{w}_{PLA}^{T} = (w_{PLA,0}, w_{PLA,1}, w_{PLA,2}, \dots, w_{PLA,d})$

$\underline{w}'^{T}_{PLA} = (w'_{PLA,0}, w'_{PLA,1}, w'_{PLA,2}, \dots, w'_{PLA,d})$

$$h(\underline{x}_n) = \text{sign}(\underline{w}_{PLA}^{T} \underline{x}_n) = \text{sign}(\underline{w}_{PLA}^{T} (1, \underline{x}_n^{orig}))$$

$$h(\underline{x}'_n) = \text{sign}(\underline{w}'^{T}_{PLA} \underline{x}'_n) = \text{sign}(\underline{w}'^{T}_{PLA} (3, 3\underline{x}_n^{orig}))$$

$$= \text{sign}(3\underline{w}'^{T}_{PLA} (1, \underline{x}_n^{orig}))$$

$$= \text{sign}(\underline{w}'^{T}_{PLA} (1, \underline{x}_n^{orig}))$$

mutiply by a positive constant won't affect the sign of the value

$\Rightarrow h(\underline{x}_n) = h(\underline{x}'_n)$

$\because$ they return the same binary classification output on every possible example in $\mathbb{R}^d$

$\therefore$ $\underline{w}_{PLA}$ and $\underline{\hat{w}}_{PLA}$ are equivalent Q.E.D.

9.

Collaborators:

B11901016 張均豪

B11901055 袁紹翔

B10502166 謝銘倫

B11901073 林禹融

From P19 in lecture 2,

upper bound $T \le \left(\frac{R}{\rho}\right)^2$,

$$R^2 = \max_n \|\underline{x}_n\|^2$$

$$= \left(\sqrt{m + 1}\right)^2 = m+1$$

$\uparrow$ $\quad\nwarrow x_0 = 1$

at most m distinct words in each article

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

$$= \text{sign}(\underline{w}_f^T \underline{x})$$

$$\begin{cases} w_{f,0} = -3.5 \\ w_{f,i} = \pm 1, \text{ for } i = 1, 2, \dots, d \end{cases}$$

$$\|\underline{w}_f\| = \sqrt{d + 3.5^2}$$

$$\rho = \min_n y_n \frac{\underline{w}_f^T}{\|\underline{w}_f\|} \underline{x}_n$$

$$= \frac{\frac{1}{2}}{\|\underline{w}_f\|}$$

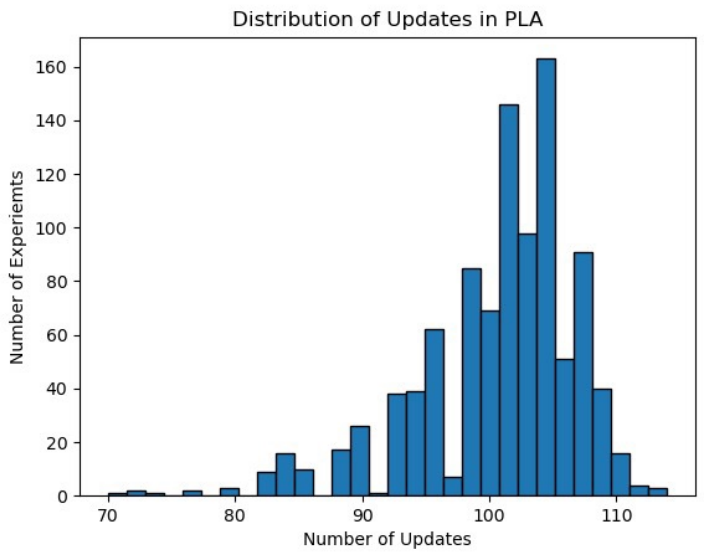
$$= \frac{\frac{1}{2}}{\frac{1}{2}\sqrt{4d + 7^2}}$$

$$= \frac{1}{\sqrt{4d+49}}$$

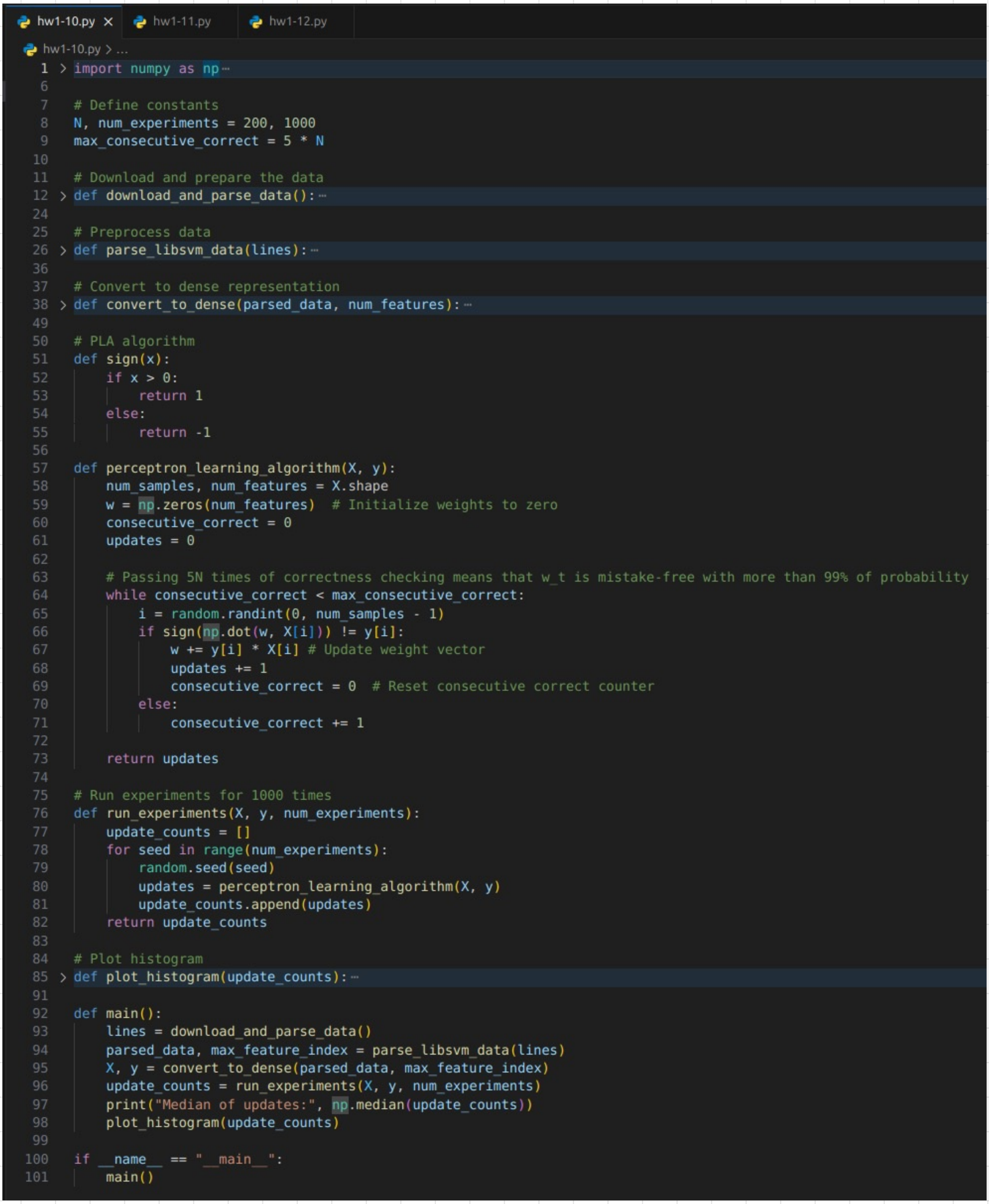
$$\therefore \quad T \le \frac{R^2}{\rho^2} = \underline{\underline{(4d+49)(m+1)}}$$

10.

Collaborators : B11901073 林禹融

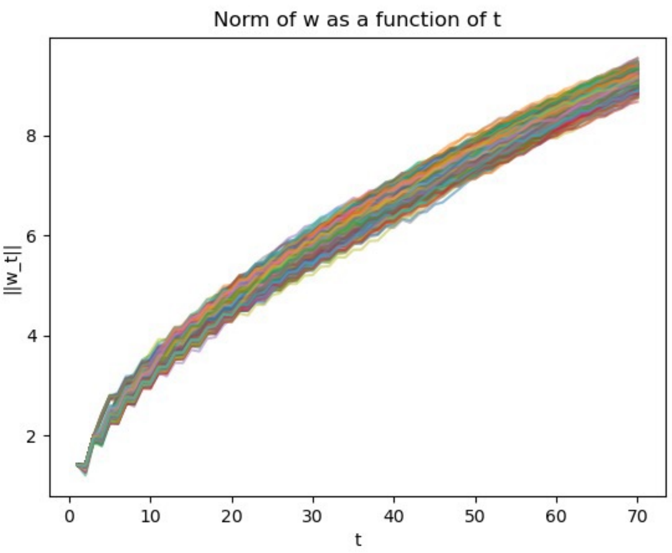


According to the histogram, most of the data concentrated between 90 and 110 updates and the median of the updates is 102.0 .

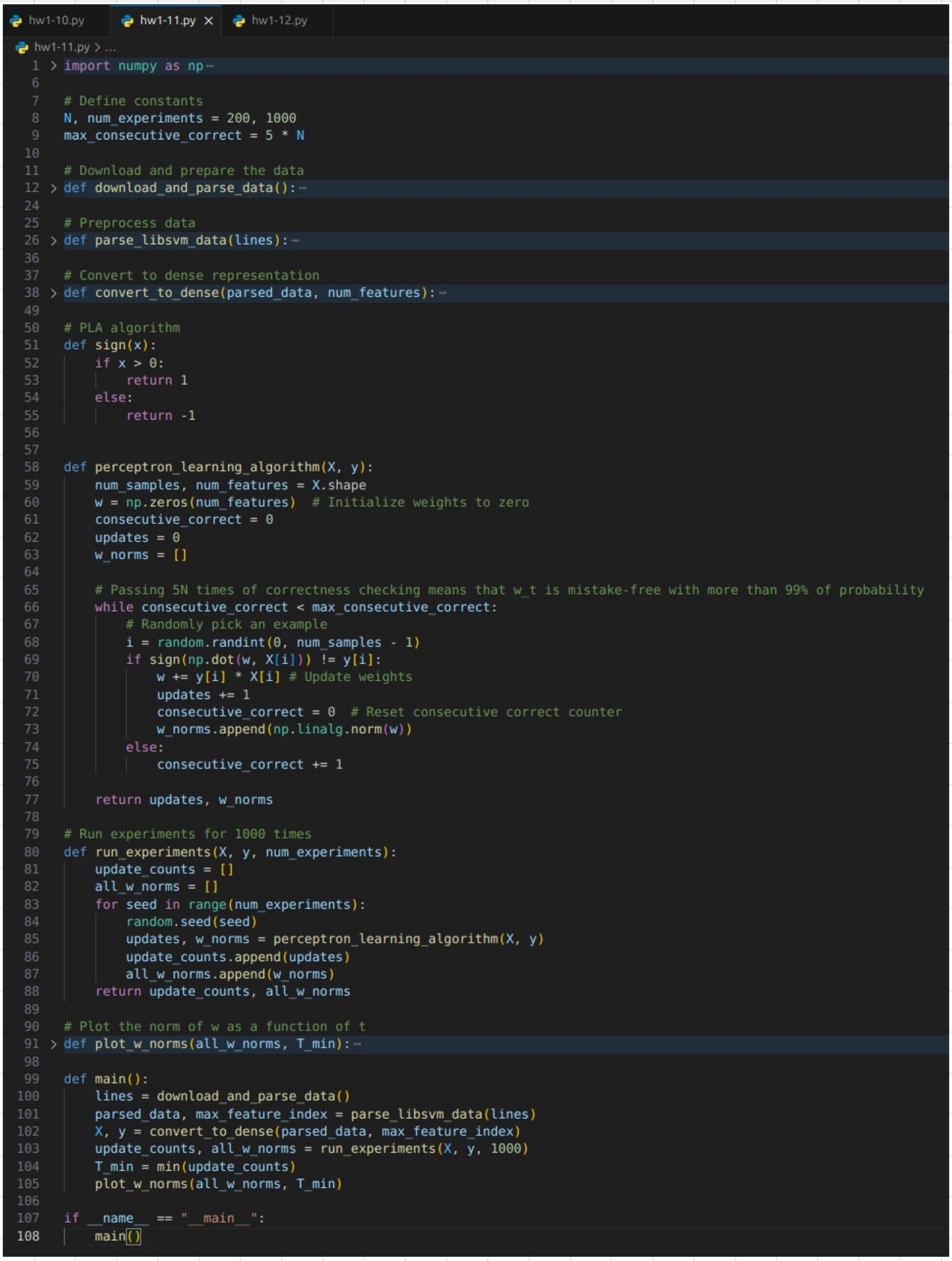


```python
import numpy as np⋯

# Define constants
N, num_experiments = 200, 1000
max_consecutive_correct = 5 * N

# Download and prepare the data
def download_and_parse_data():⋯

# Preprocess data
def parse_libsvm_data(lines):⋯

# Convert to dense representation
def convert_to_dense(parsed_data, num_features):⋯

# PLA algorithm
def sign(x):
    if x > 0:
        return 1
    else:
        return -1

def perceptron_learning_algorithm(X, y):
    num_samples, num_features = X.shape
    w = np.zeros(num_features)  # Initialize weights to zero
    consecutive_correct = 0
    updates = 0

    # Passing 5N times of correctness checking means that w_t is mistake-free with more than 99% of probability
    while consecutive_correct < max_consecutive_correct:
        i = random.randint(0, num_samples - 1)
        if sign(np.dot(w, X[i])) != y[i]:
            w += y[i] * X[i] # Update weight vector
            updates += 1
            consecutive_correct = 0  # Reset consecutive correct counter
        else:
            consecutive_correct += 1

    return updates

# Run experiments for 1000 times
def run_experiments(X, y, num_experiments):
    update_counts = []
    for seed in range(num_experiments):
        random.seed(seed)
        updates = perceptron_learning_algorithm(X, y)
        update_counts.append(updates)
    return update_counts

# Plot histogram
def plot_histogram(update_counts):⋯

def main():
    lines = download_and_parse_data()
    parsed_data, max_feature_index = parse_libsvm_data(lines)
    X, y = convert_to_dense(parsed_data, max_feature_index)
    update_counts = run_experiments(X, y, num_experiments)
    print("Median of updates:", np.median(update_counts))
    plot_histogram(update_counts)

if __name__ == "__main__":
    main()
```

11. no collaborators

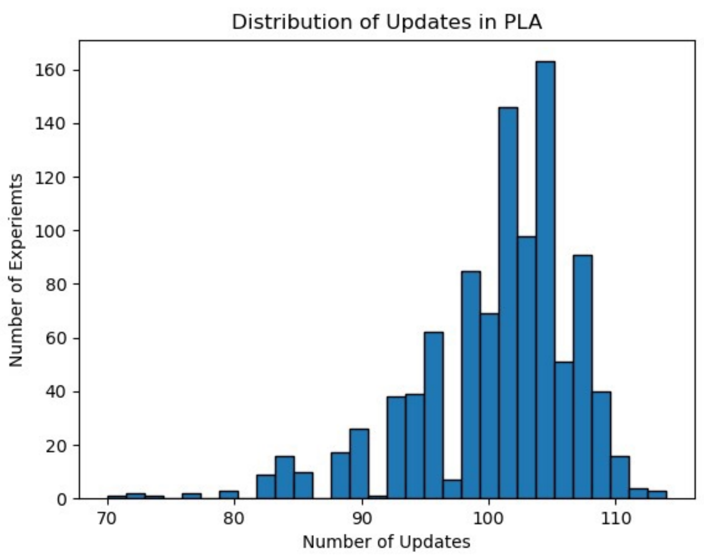


According to the figure, all the functions increase steadily over t, indicating that $\|\underline{w_t}\|$ grows slowly. Furthermore, the curves grow zig-zag, reflecting small differences in some updates in different experiements before PLA algorithm converges.

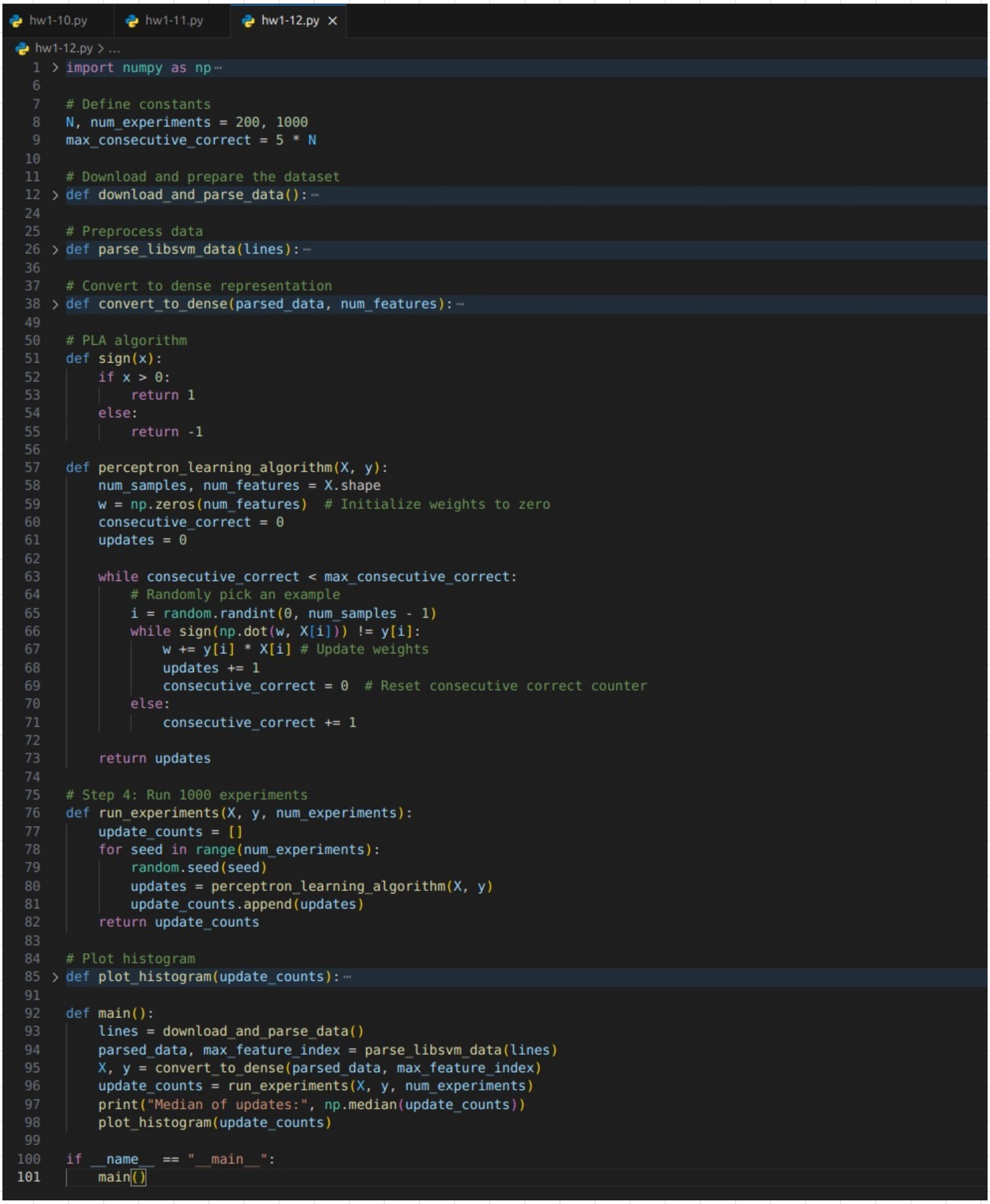


```python
import numpy as np ⋯

# Define constants
N, num_experiments = 200, 1000
max_consecutive_correct = 5 * N

# Download and prepare the data
def download_and_parse_data(): ⋯

# Preprocess data
def parse_libsvm_data(lines): ⋯

# Convert to dense representation
def convert_to_dense(parsed_data, num_features): ⋯

# PLA algorithm
def sign(x):
    if x > 0:
        return 1
    else:
        return -1


def perceptron_learning_algorithm(X, y):
    num_samples, num_features = X.shape
    w = np.zeros(num_features)  # Initialize weights to zero
    consecutive_correct = 0
    updates = 0
    w_norms = []

    # Passing 5N times of correctness checking means that w_t is mistake-free with more than 99% of probability
    while consecutive_correct < max_consecutive_correct:
        # Randomly pick an example
        i = random.randint(0, num_samples - 1)
        if sign(np.dot(w, X[i])) != y[i]:
            w += y[i] * X[i] # Update weights
            updates += 1
            consecutive_correct = 0  # Reset consecutive correct counter
            w_norms.append(np.linalg.norm(w))
        else:
            consecutive_correct += 1

    return updates, w_norms

# Run experiments for 1000 times
def run_experiments(X, y, num_experiments):
    update_counts = []
    all_w_norms = []
    for seed in range(num_experiments):
        random.seed(seed)
        updates, w_norms = perceptron_learning_algorithm(X, y)
        update_counts.append(updates)
        all_w_norms.append(w_norms)
    return update_counts, all_w_norms

# Plot the norm of w as a function of t
def plot_w_norms(all_w_norms, T_min): ⋯

def main():
    lines = download_and_parse_data()
    parsed_data, max_feature_index = parse_libsvm_data(lines)
    X, y = convert_to_dense(parsed_data, max_feature_index)
    update_counts, all_w_norms = run_experiments(X, y, 1000)
    T_min = min(update_counts)
    plot_w_norms(all_w_norms, T_min)

if __name__ == "__main__":
    main()
```

12. no collaborators



The median of updates is 102.0 which is equal to that in problem 10. The histogram in promble 12 and 10 also look alike.



```python
import numpy as np⋯

# Define constants
N, num_experiments = 200, 1000
max_consecutive_correct = 5 * N

# Download and prepare the dataset
def download_and_parse_data():⋯

# Preprocess data
def parse_libsvm_data(lines):⋯

# Convert to dense representation
def convert_to_dense(parsed_data, num_features):⋯

# PLA algorithm
def sign(x):
    if x > 0:
        return 1
    else:
        return -1

def perceptron_learning_algorithm(X, y):
    num_samples, num_features = X.shape
    w = np.zeros(num_features)  # Initialize weights to zero
    consecutive_correct = 0
    updates = 0

    while consecutive_correct < max_consecutive_correct:
        # Randomly pick an example
        i = random.randint(0, num_samples - 1)
        while sign(np.dot(w, X[i])) != y[i]:
            w += y[i] * X[i] # Update weights
            updates += 1
            consecutive_correct = 0  # Reset consecutive correct counter
        else:
            consecutive_correct += 1

    return updates

# Step 4: Run 1000 experiments
def run_experiments(X, y, num_experiments):
    update_counts = []
    for seed in range(num_experiments):
        random.seed(seed)
        updates = perceptron_learning_algorithm(X, y)
        update_counts.append(updates)
    return update_counts

# Plot histogram
def plot_histogram(update_counts):⋯

def main():
    lines = download_and_parse_data()
    parsed_data, max_feature_index = parse_libsvm_data(lines)
    X, y = convert_to_dense(parsed_data, max_feature_index)
    update_counts = run_experiments(X, y, num_experiments)
    print("Median of updates:", np.median(update_counts))
    plot_histogram(update_counts)

if __name__ == "__main__":
    main()
```

13. ① prove that $\underline{w_{t+1}}$ always correctly classifies $(\underline{x_{n(t)}}, y_{n(t)})$ after the update

$\Leftrightarrow y_{n(t)} \underline{w_{t+1}}^T \underline{x_{n(t)}} > 0$

no collaborators

pf. Known that $y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}} < 0$

$y_{n(t)} \underline{w_{t+1}}^T \underline{x_{n(t)}}$

$$= y_{n(t)} \left(\underline{w_t} + \frac{1}{10} y_{n(t)} \underline{x_{n(t)}} \cdot \left\lfloor \frac{-10\, y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}}}{\|\underline{x_{n(t)}}\|^2} + 1 \right\rfloor \right)^T \underline{x_{n(t)}}$$

$$= y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}} + \frac{1}{10} y_{n(t)}^2 \|\underline{x_{n(t)}}\|^2 \left\lfloor \frac{-10\, y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}}}{\|\underline{x_{n(t)}}\|^2} + 1 \right\rfloor$$

$$> y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}} + \frac{1}{10} \|\underline{x_{n(t)}}\|^2 \left(\frac{-10\, y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}}}{\|\underline{x_{n(t)}}\|^2} + 1 - 1 \right)$$

$$= y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}} - y_{n(t)} \underline{w_t}^T \underline{x_{n(t)}}$$

$$= 0$$

$\therefore y_{n(t)} \underline{w_{t+1}}^T \underline{x_{n(t)}} > 0$ Q.E.D.

② prove : the data is linearly separable

$\Rightarrow$ such a PLA halts with a perfect hyperplane

pf. $\because$ the data is linearly separable

$\therefore$ there exists a weight vector $\underline{w_f}$ defining a hyperplane that can perfectly classify the data

$\Rightarrow y_{n(t)} \underline{w_f}^T \underline{x_{n(t)}} > 0 \quad \forall n$

from ①, $\underline{w_{t+1}}$ always correctly classifies $(\underline{x_{n(t)}}, y_{n(t)})$ after the update

$\Rightarrow$ the number of misclassified points will decrease and eventually reaching zero when the weight vector be the $\underline{w_f}$

$\Rightarrow$ such a PLA haults with a perfect hyperplane defined by $\underline{w_f}$ Q.E.D.