Problem 5

ChatGPT says:
A notable application of active learning in machine learning is in data labeling for supervised learning tasks.
Example: Data Labeling with Active Learning
In this scenario, an active learning algorithm selectively queries a human annotator to label the most informative data points from a large, unlabeled dataset. Here is how it works:

1. Initial Model Training: A preliminary model is trained on a small labeled dataset.

2. Uncertainty Sampling:The model identifies which unlabeled data points it is least confident about (e.g. those near the decision boundary).

3. Querying for Labels: The model then queries the human annotator to label these uncertain data points, rather than labeling the entire dataset.

4. Model Improvement: The newly labeled data is added to the training set, and the model is retrained to improve its accuracy.

5. Iterative Process: This process is repeated, with the model continuously refining its predictions and focusing on the most challenging samples.

By using active learning in this way, the model can achieve high performance with significantly less labeled data, making it especially useful in situations where labeling is costly or time-consuming.

My arguement:
Active learning focus on question asking. The example supplied by ChatGPT is good. It is a standard application of active learning. Since the data we want to label would be a large amount, we cannot label them manually. We can first label a few numbers of the data to let machine learn, then let machine help us to label the data which we have not labeled. In the labeling process, the agent will ask us the data it has less confidence. From the asking, the agent can label the data more correctly.

Problem 6

ChatGPT says:
Machine learning can be utilized to analyze patterns in seismic data and potentially improve earthquake prediction efforts, though it's important to note that accurately predicting earthquakes remains a complex challenge.

1. How Machine Learning Can Help: Seismic Data Analysis: Machine learning algorithms can analyze vast amounts of seismic data from sensors and historical earthquake records to identify patterns and anomalies that may precede seismic events.

2. Feature Extraction: Algorithms can extract relevant features from raw seismic signals, such as frequency patterns, amplitude changes, or other geophysical characteristics that might indicate an impending quake.

3. Classification Models: Supervised learning models can be trained to classify seismic events, distinguishing between typical seismic noise and potentially significant tremors.

4. Anomaly Detection: Unsupervised learning techniques can help identify unusual patterns in seismic data that may signal an increased risk of an earthquake.

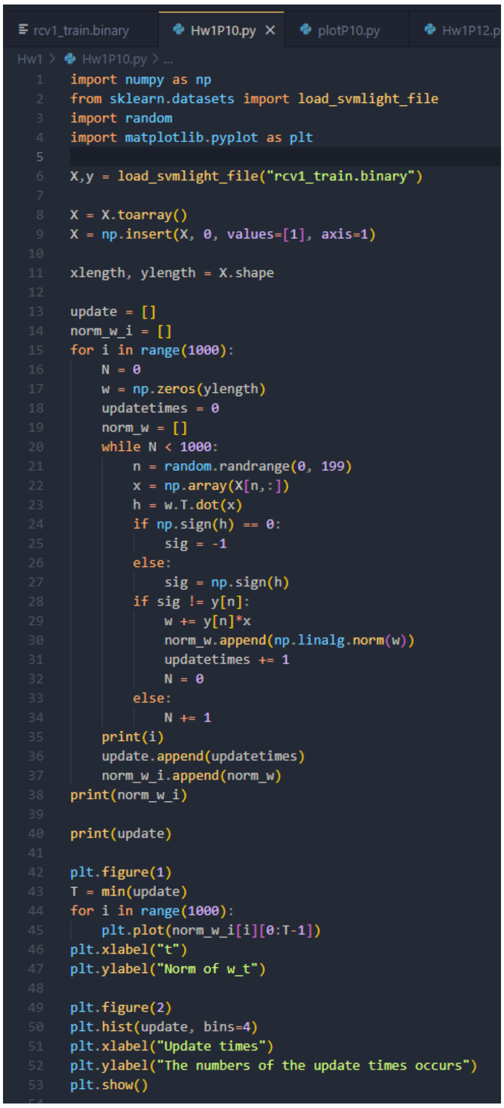
5. Real-Time Monitoring: Machine learning can enhance real-time monitoring systems by providing quicker assessments of seismic activity and improving alert systems for potential earthquakes.

Limitations: While machine learning offers valuable tools for analyzing seismic data, accurately predicting the exact time, location, and magnitude of earthquakes is still beyond current capabilities. Earthquakes are influenced by complex geological processes that are not fully understood, making precise predictions difficult. However, machine learning can contribute to better understanding seismic risks and enhancing preparedness.

My arguement:
The answer from ChatGPT is very useful. Since we have not analyzed the reason why earthquake occurs completely, we can only recognize earthquake by the features we have already known. Machine learning is a great tool to help us analyze the data we collected from the happened earthquake. This question is the question we want to find x from getting y. By using machine learning, we can find the relativity between any features we've collected and the earthquake. ChatGPT gives us a great and detailed.

Problem 10

We can observe that the distribution of the update times of the weight $\boldsymbol{w}$ is concentrated around 100. Most of them are in the interval $[95, 105]$, which is about $\frac{N}{2}$. By the figure 1, we can find out that the update times is about half of the numbers of the data.




```python
import numpy as np
from sklearn.datasets import load_svmlight_file
import random
import matplotlib.pyplot as plt

X,y = load_svmlight_file("rcv1_train.binary")

X = X.toarray()
X = np.insert(X, 0, values=[1], axis=1)

xlength, ylength = X.shape

update = []
norm_w_i = []
for i in range(1000):
    N = 0
    w = np.zeros(ylength)
    updatetimes = 0
    norm_w = []
    while N < 1000:
        n = random.randrange(0, 199)
        x = np.array(X[n,:])
        h = w.T.dot(x)
        if np.sign(h) == 0:
            sig = -1
        else:
            sig = np.sign(h)
        if sig != y[n]:
            w += y[n]*x
            norm_w.append(np.linalg.norm(w))
            updatetimes += 1
            N = 0
        else:
            N += 1
    print(i)
    update.append(updatetimes)
    norm_w_i.append(norm_w)
print(norm_w_i)

print(update)

plt.figure(1)
T = min(update)
for i in range(1000):
    plt.plot(norm_w_i[i][0:T-1])
plt.xlabel("t")
plt.ylabel("Norm of w_t")

plt.figure(2)
plt.hist(update, bins=4)
plt.xlabel("Update times")
plt.ylabel("The numbers of the update times occurs")
plt.show()
```

Figure 1: snapshot

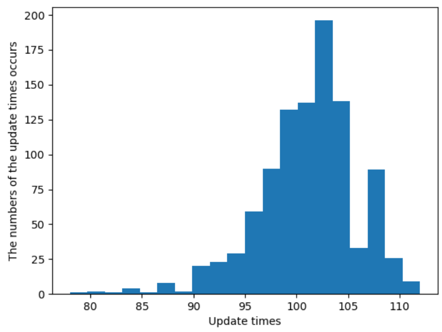


Figure 2: historgram

Problem 11


```python
import numpy as np
from sklearn.datasets import load_svmlight_file
import random
import matplotlib.pyplot as plt

X,y = load_svmlight_file("rcv1_train.binary")

X = X.toarray()
X = np.insert(X, 0, values=[1], axis=1)

xlength, ylength = X.shape

update = []
norm_w_i = []
for i in range(1000):
    N = 0
    w = np.zeros(ylength)
    updatetimes = 0
    norm_w = []
    while N < 1000:
        n = random.randrange(0, 199)
        x = np.array(X[n,:])
        h = w.T.dot(x)
        if np.sign(h) == 0:
            sig = -1
        else:
            sig = np.sign(h)
        if sig != y[n]:
            w += y[n]*x
            norm_w.append(np.linalg.norm(w))
            updatetimes += 1
            N = 0
        else:
            N += 1
    print(i)
    update.append(updatetimes)
    norm_w_i.append(norm_w)
print(norm_w_i)

print(update)

plt.figure(1)
T = min(update)
for i in range(1000):
    plt.plot(norm_w_i[i][0:T-1])
plt.xlabel("t")
plt.ylabel("Norm of w_t")

plt.figure(2)
plt.hist(update, bins=4)
plt.xlabel("Update times")
plt.ylabel("The numbers of the update times occurs")
plt.show()
```

Figure 3: snapshot

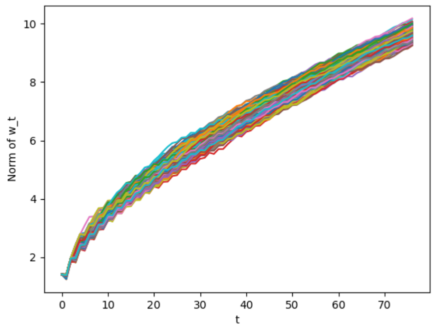
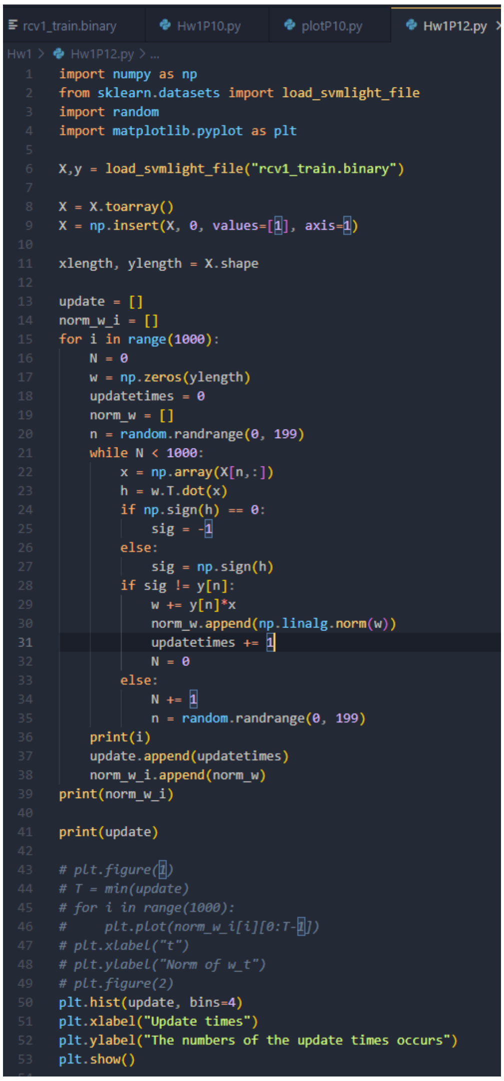


Figure 4: plot

Problem 12

For the P12, we update $n(t)$ only when $\boldsymbol{w}_t$ does not change. Compare with Figure 2 we can observe that most of the update times are still in $[95, 105]$. But it looks more concentrated then the histogram in P10. That is, the update times are more stable than P10.



```python
import numpy as np
from sklearn.datasets import load_svmlight_file
import random
import matplotlib.pyplot as plt

X,y = load_svmlight_file("rcv1_train.binary")

X = X.toarray()
X = np.insert(X, 0, values=[1], axis=1)

xlength, ylength = X.shape

update = []
norm_w_i = []
for i in range(1000):
    N = 0
    w = np.zeros(ylength)
    updatetimes = 0
    norm_w = []
    n = random.randrange(0, 199)
    while N < 1000:
        x = np.array(X[n,:])
        h = w.T.dot(x)
        if np.sign(h) == 0:
            sig = -1
        else:
            sig = np.sign(h)
        if sig != y[n]:
            w += y[n]*x
            norm_w.append(np.linalg.norm(w))
            updatetimes += 1
            N = 0
        else:
            N += 1
            n = random.randrange(0, 199)
    print(i)
    update.append(updatetimes)
    norm_w_i.append(norm_w)
print(norm_w_i)

print(update)

# plt.figure(1)
# T = min(update)
# for i in range(1000):
#     plt.plot(norm_w_i[i][0:T-1])
# plt.xlabel("t")
# plt.ylabel("Norm of w_t")
# plt.figure(2)
plt.hist(update, bins=4)
plt.xlabel("Update times")
plt.ylabel("The numbers of the update times occurs")
plt.show()
```

Figure 5: snapshot

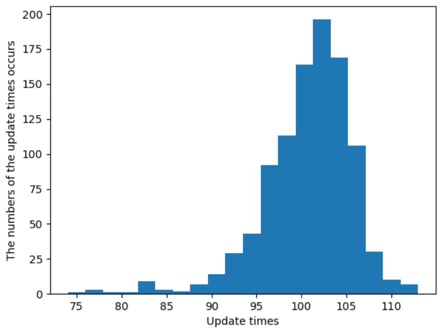


Figure 6: historgram