# Project 4 Report

Xinyu Cai, Ningyuan Xiong

## Instructions to run the project:

For linear classifier part, we have four main classes: PerceptronGraphMain, LogisticGraphMain, PerceptronMain, and LogisticMain. The first two main classes show the result and the graph, while the last two main classes only show the result. Instruction of how to run the project are below (no need to enter the file name, we coded the filename inside the methods):

Instructions to compile:

```
javac learn/lc/examples/PerceptronGraphMain.java
javac learn/lc/examples/PerceptronMain.java
javac learn/lc/examples/LogisticGraphMain.java
javac learn/lc/examples/LogisticMain.java
```

Our default number of steps and alpha are 10k (100k for Earthquake Noisy decaying) and 0.95. If you want to change the value, simply add the numbers separated by space after each instruction.

If you just want the graph, then

```
java learn/lc/examples/PerceptronGraphMain EarthquakeClean
java learn/lc/examples/PerceptronGraphMain EarthquakeNoisy
java learn/lc/examples/PerceptronGraphMain EarthquakeNoisyDecaying
java learn/lc/examples/PerceptronGraphMain HouseVotes

java learn/lc/examples/LogisticGraphMain EarthquakeClean
java learn/lc/examples/LogisticGraphMain EarthquakeNoisy
java learn/lc/examples/LogisticGraphMain EarthquakeNoisyDecaying
java learn/lc/examples/LogisticGraphMain HouseVotes
```

If you only want to show the data (take earthquake clean as an example), then

```
java learn/lc/examples/PerceptronMain EarthquakeClean
java learn/lc/examples/LogisticMain EarthquakeClean
```

For neural network, we have two main files: IrisNeuralNetwork and MNISTNeuralNetwork.

Instructions to compile:

```
javac learn/nn/examples/IrisNeuralNetwork.java
javac learn/nn/examples/MNISTNeuralNetwork.java
```

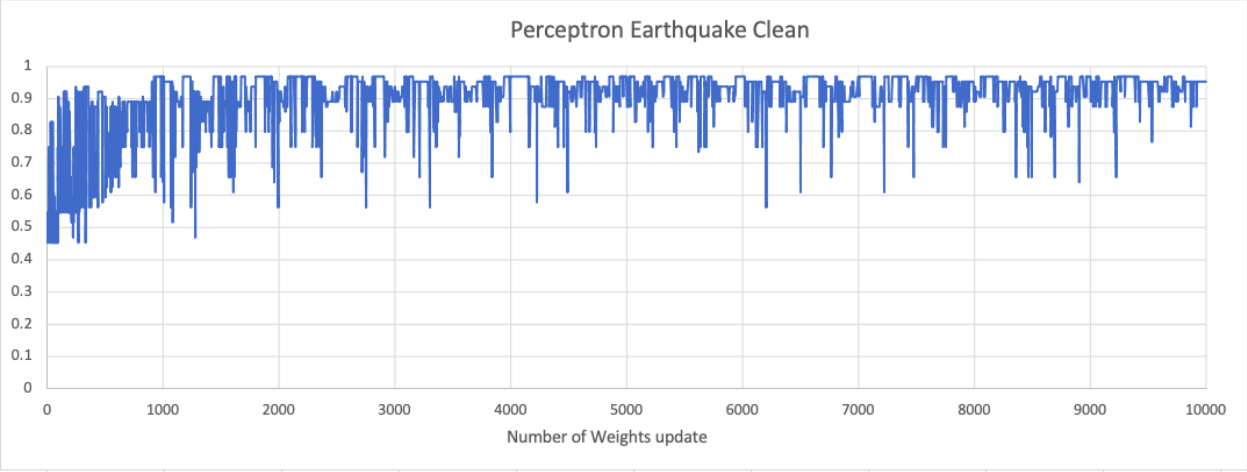
To run:

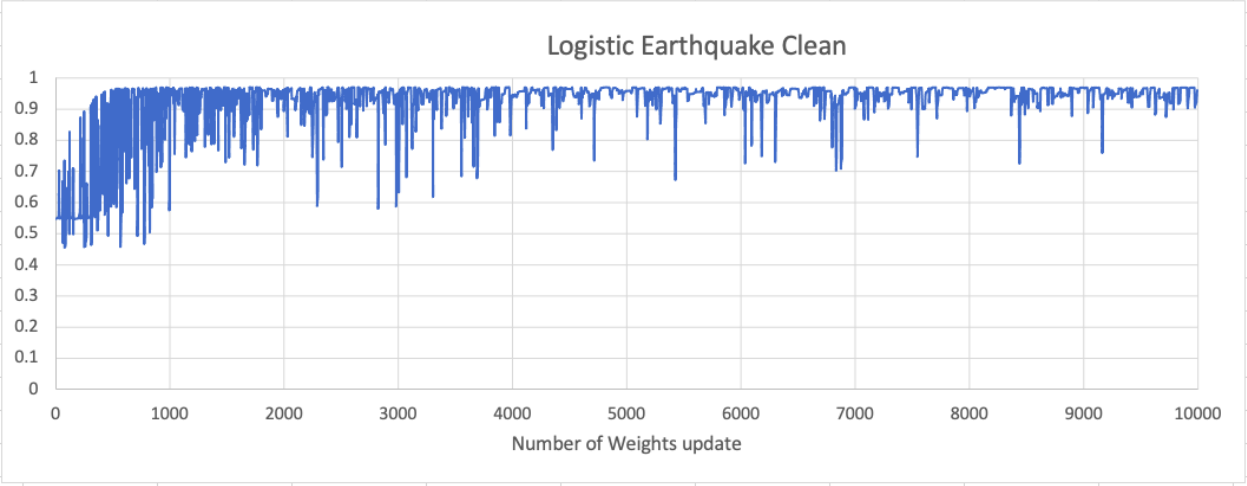
```
java learn/nn/examples/IrisNeuralNetwork
java learn/nn/examples/MNISTNeuralNetwork
```

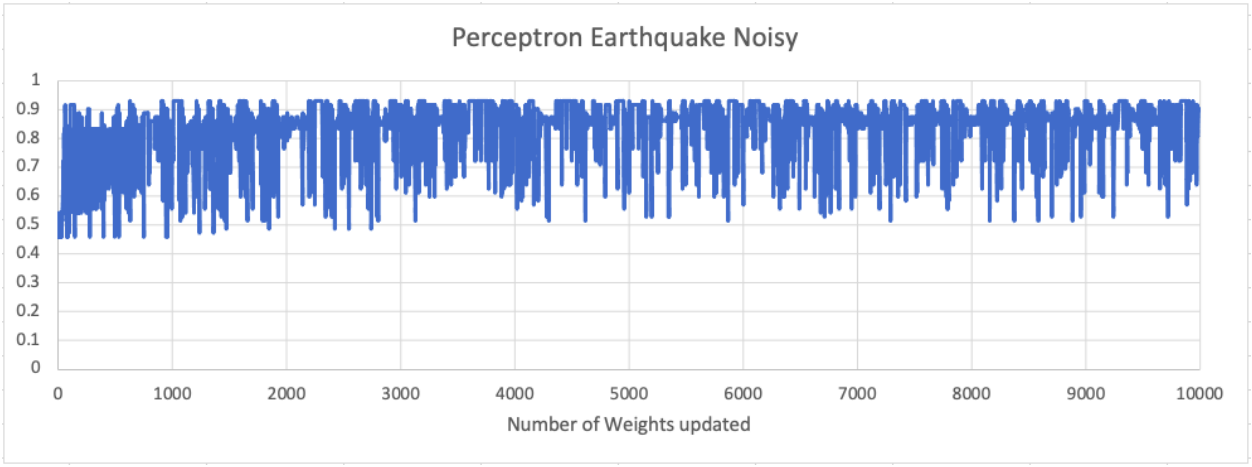
Description:
We implemented both the logistic classifiers and the perceptron classifier. These two classifiers both work perfectly on the earthquake data and the “house votes” dataset. We also tried to implement the neural network, and it can work on Iris dataset MNIST handwriting database.

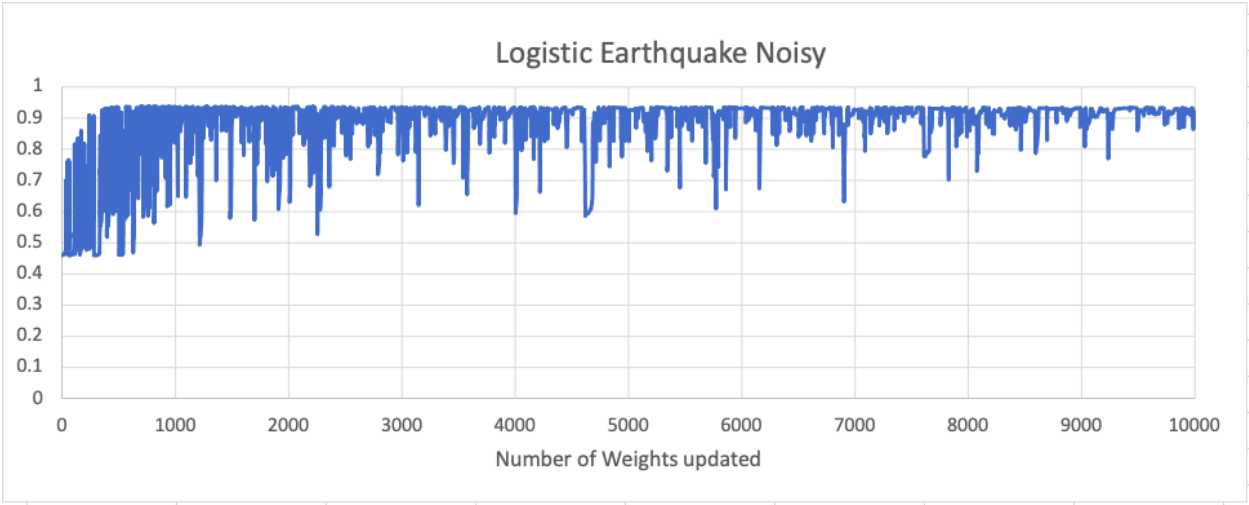
Summary:
In general, logistic classifier works more accurately than perceptron classifier especially on noisy data. The learning curves on our graphs showed that perceptron classifier have higher variability, while the logistic classifier have less variability and more accuracy. The result of this large difference is because these two classifiers use two different threshold function.

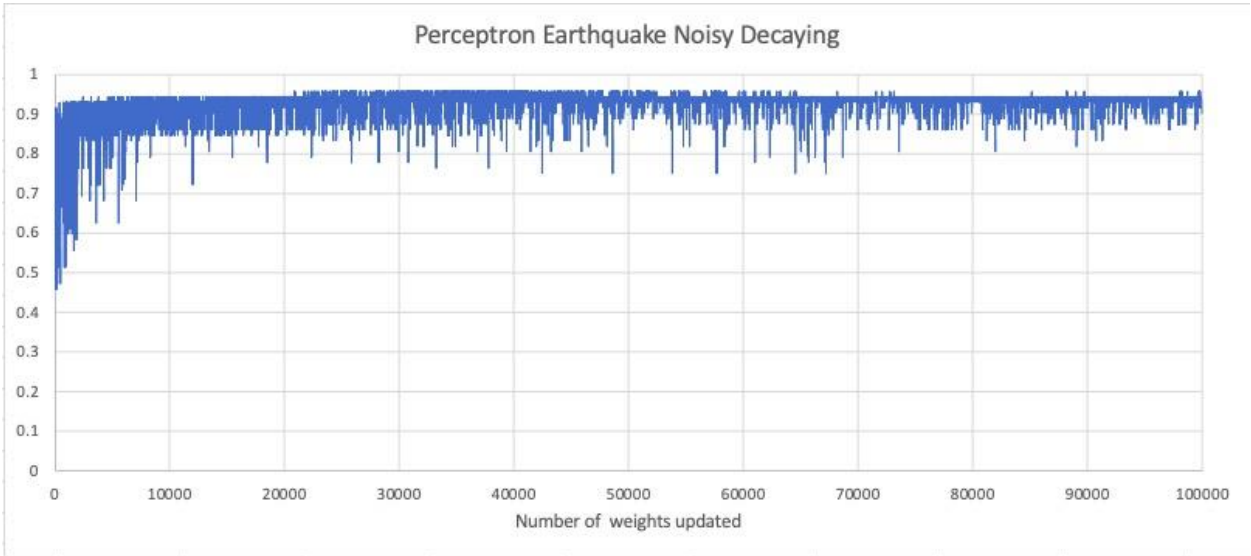
Graphs:
All x-axis ranges from 0 to 10000 (earthquake noisy decay ranges from 0 to 100000).
Y-axis ranges from 0 to 1 (logistic house votes ranges from 0.86 to 1.0).

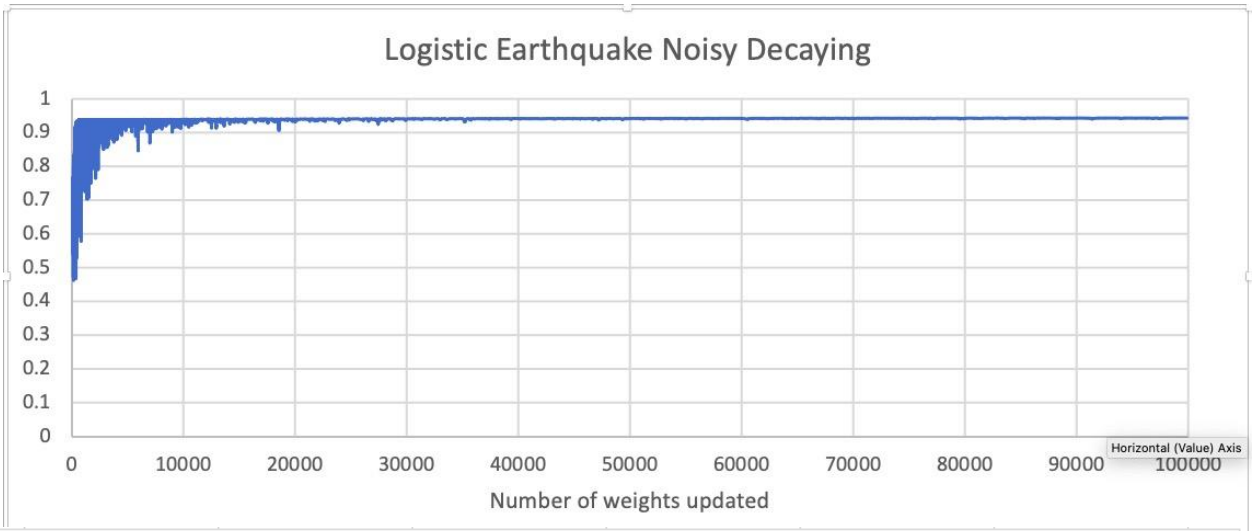


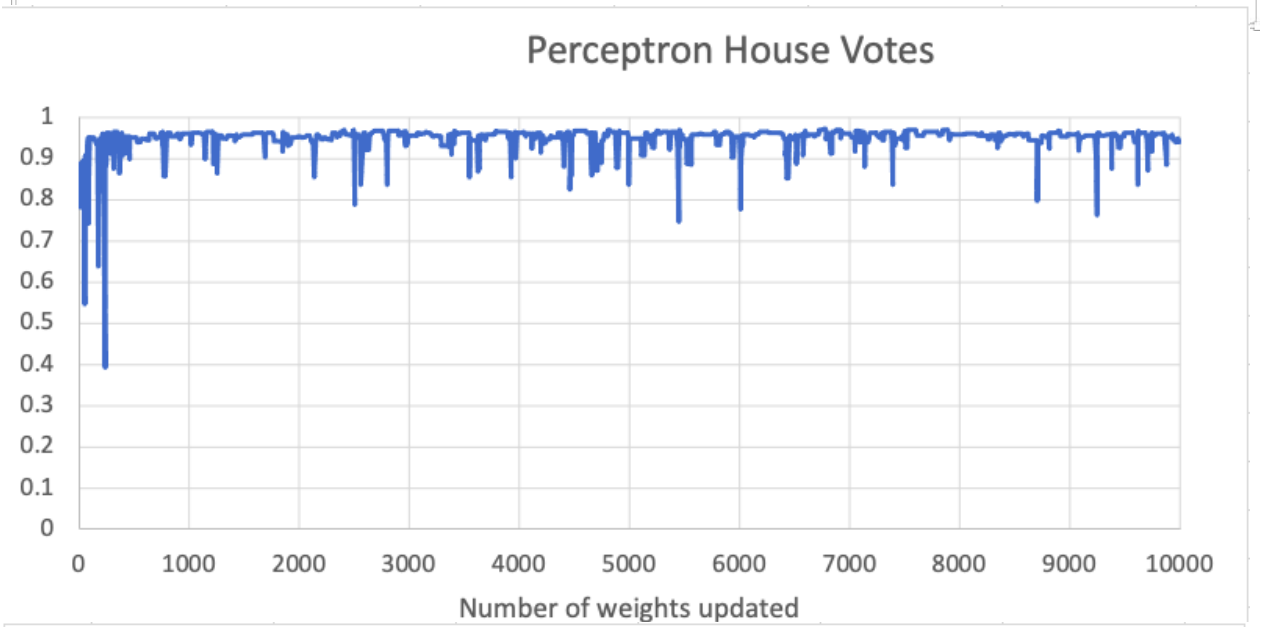



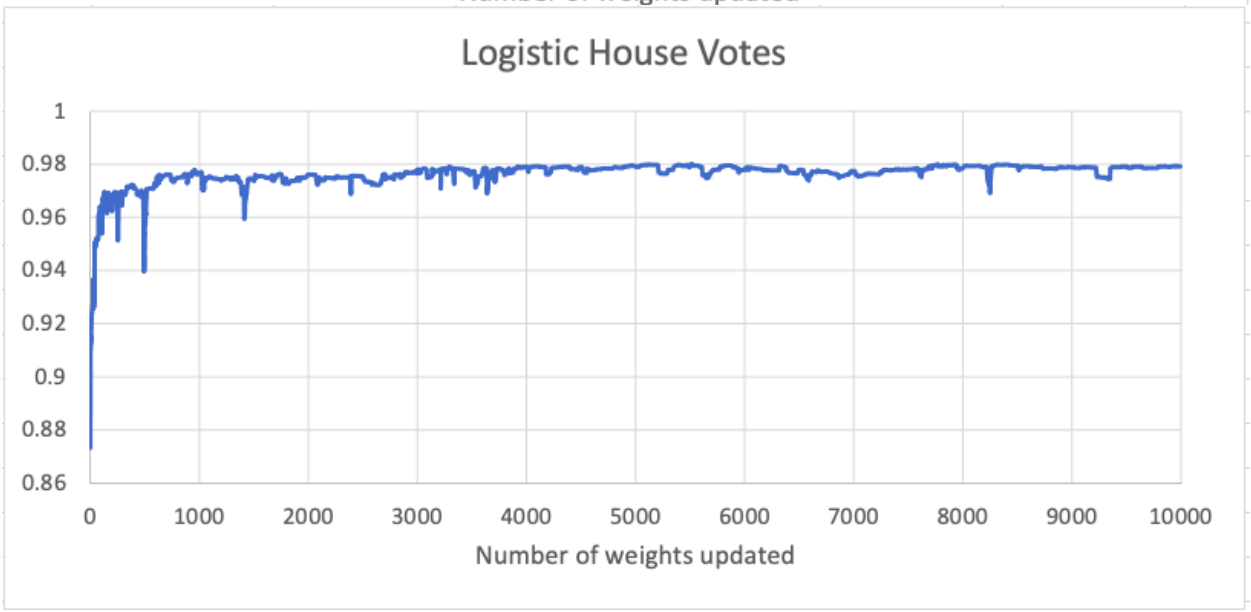

Perceptron Earthquake Noisy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Number of Weights updated


Logistic Earthquake Noisy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Number of Weights updated


Perceptron Earthquake Noisy Decaying
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of weights updated


Logistic Earthquake Noisy Decaying
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Horizontal (Value) Axis
Number of weights updated

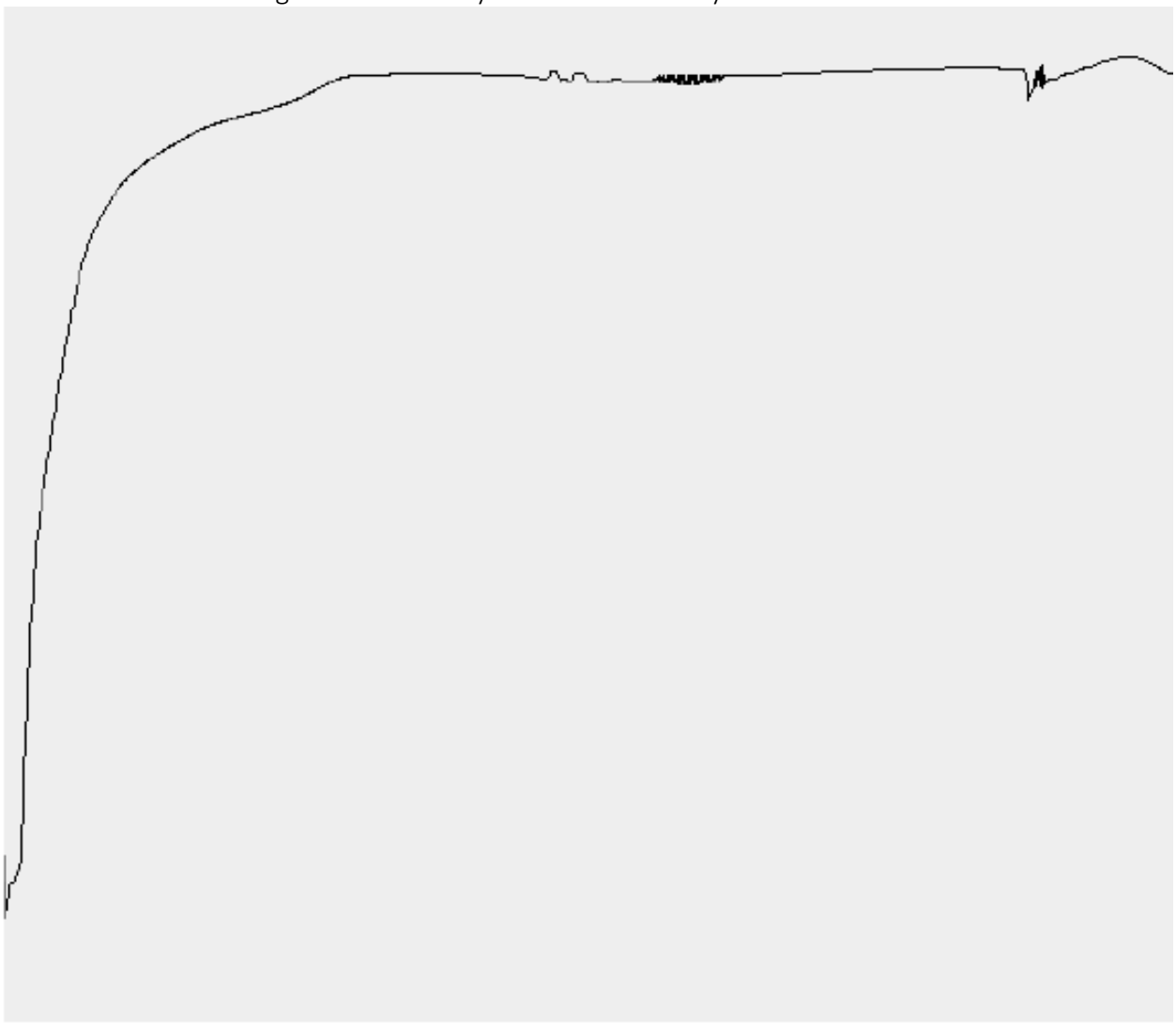

Perceptron House Votes
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Number of weights updated


Logistic House Votes
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Number of weights updated

Neural Network

Iris Dataset

The x axis is the training round and the y axis is the accuracy for each around



Although we implemented MNISTNN, it took too much time.