# Homework 1

Xinyu Liu

September 2021

## Analytical Part

### 1.

Let's assume that we have 3 weights for voted and averaged perceptron. We also give each weight count 1. So we have $w_1, w_2, w_3$, with $c_1 = c_2 = c_3 = 1$.

a. Voted perceptron would be $f(x) = Majority\ of\ Sign(< w1, x >, < w2, x >, < w3, x >)$ . As we can see, the equation takes the common space made by lines $w_1, w_2, w_3$. Therefore, it's not linear.

b. Averaged perceptron in this case would be $f(x) = Sign(w_1 + w_2 + w_3, x)$. The decision boundary is combined by all three weights and their count. We can transform it to $f(x) = Sign((w_0 + y_0x_0) + (w_1 + y_1x_1) + (w_2 + y_2x_2))$. The result is a vector, thus it's linear.

### 2.

In order to keep margin of 1, original equation is

$$w_{t+1} = \min_w \frac{1}{2}\|w - w_t\|^2 \quad s.t. y_t(w \cdot x_t) \geq 1$$

Instead of margin 1, we substitute it with margin M, we then have

$$w_{t+1} = \min_w \frac{1}{2}\|w - w_t\|^2 \quad s.t. y_t(w \cdot x_t) \geq M$$

The Lagrangian optimization becomes:

$$L(w, \tau) = \frac{1}{2}\|w - w_t\|^2 + \tau(M - y_t(w \cdot x_t))$$

$$\frac{\partial L}{\partial w} = (w - w_t) - \tau y_t x_t$$

L global optimum when $\frac{\partial L}{\partial w} = 0$, therefore

$$0 = (w - w_t) - \tau y_t x_t$$

$$w = w_t + \tau y_t x_t$$

The original Lagrangian optimization looks like this with optimized $w$

$$L(\tau) = -\frac{1}{2}\|x_t\|^2\tau^2 + \tau(M - y_t(w_t \cdot x_t))$$

$$\tau = max\{0, \frac{M - y_t(w_t \cdot x_t)}{\|x_t\|^2}\}$$

Put learning rate $\tau$ back to the weight update function, we then get:

$$w_{t+1} = w_t + \frac{M - y_t(w_t \cdot x_t)}{\|x_t\|^2} y_t x_t$$

### 3.

#### a

In standard perceptron, we have $w_{t+1} = w_t + y_t x_t$. When an importance score $h_t$ is provided, we can multiply the result $y_t$ by $h_t$, which then becomes $w_{t+1} = w_t + h_t y_t x_t$.

#### b

Using answers from part a. we still use the strategy of multiplying importance score $h_t$ to result $y_t$. In standard perceptron, $h_t$ is always 1. Without changing the standard perceptron method, we have to reduce the problem. We can write

$$w_{t+1} = w_t + h_t y_t x_t$$

Instead of changing learning rate like part a. did, we can multiply the original dataset result by $h_t$. For example, the orignal dataset $(x_0, y_0, h_0)$ has corresponding values (0, 1, 10). We can modify the $y_0$ result value and remove $h_0$ by making $y_0' = y_0 \cdot h_0 = 1 \cdot 10 = 10$.

### 4.

#### a

Since we care more about positive accuracy, we can deal with the problem by giving positive result more importance score like problem 3 did. For example, we consider positive examples are 5 times more important than negative scores, then $\forall y_t > 0, h_t = 5$, else $h_t = 1$.

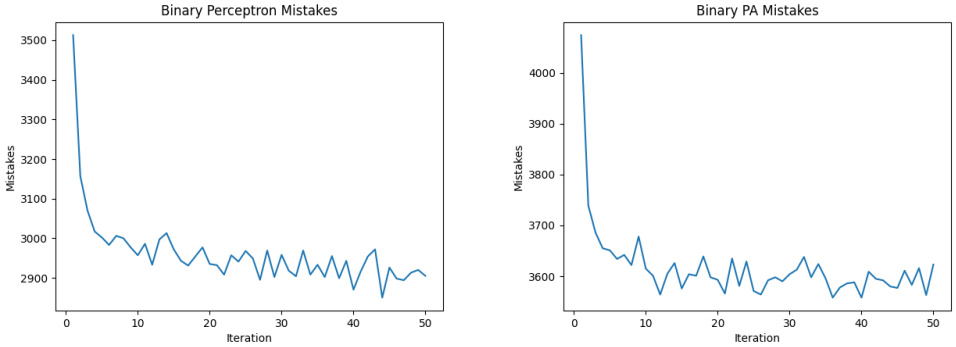
#### b

Just like problem 3b, instead of using a weight factor $h_t$, we modify the input $y_t$ by multiplying with $h_t$ when $y_t > 0$ upon reading data, thus still remain integrity of original perceptron $w_{t+1} = w_t + y_t x_t$

## Programming and Empirical Analysis Part

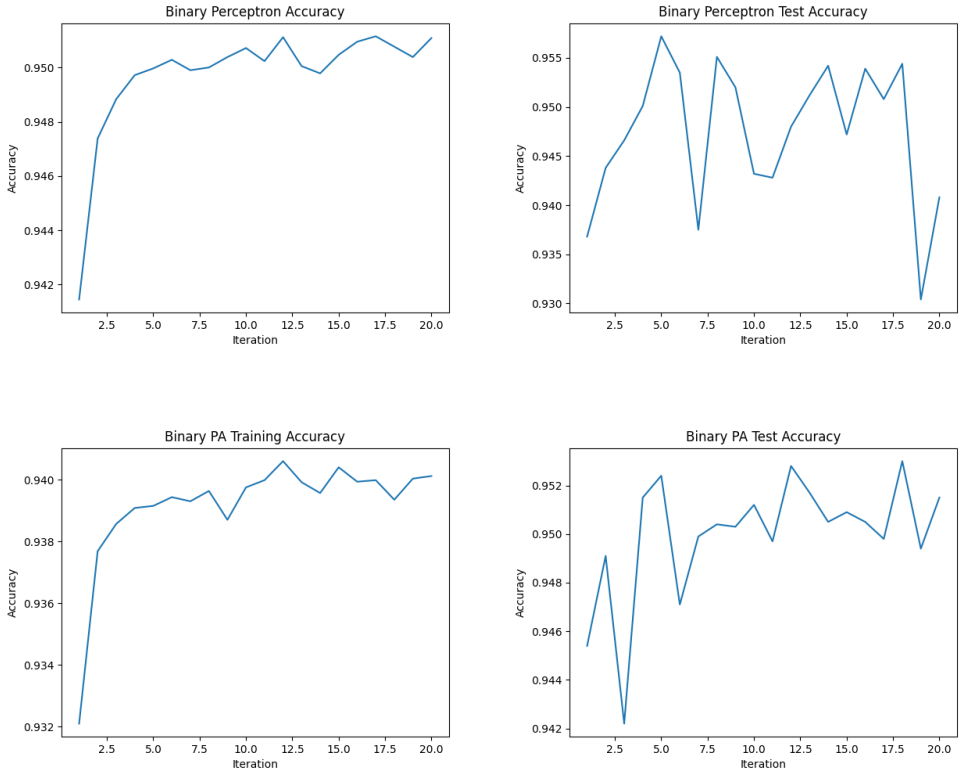
### 1.

#### a



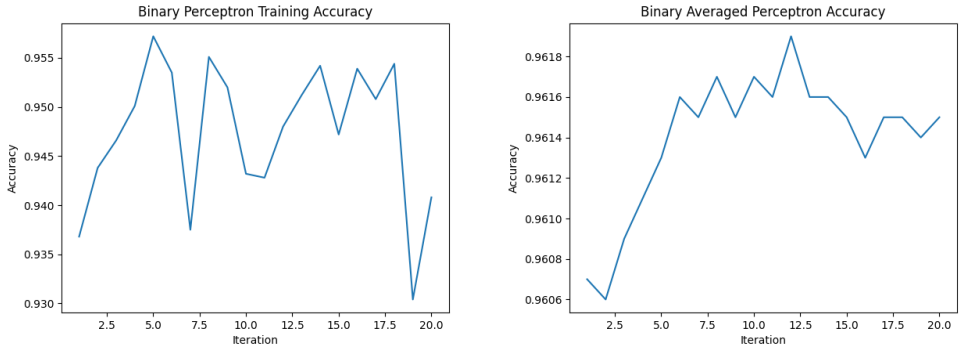

As we can see from the graph, perceptron has lower mistakes than PA. Both graph show that the algorithm hits optimum at roughly 10 iterations.

**b**

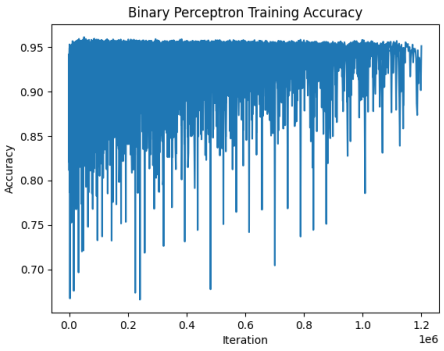


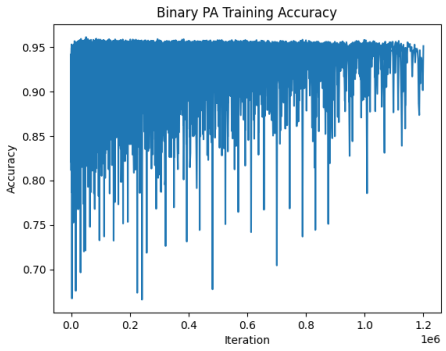

Perceptron has higher accuracy in training set, but PA has higher overall accuracy in test set. Also, PA's deviation of accuracy becomes narrower as iteration goes further.

**c**




According to the graphs, the lowest accuracy of averaged perceptron is higher than the highest of standard perceptron. Also, averaged perceptron reaches optimal after about 5 iterations whereas standard perceptron has no optimal point.



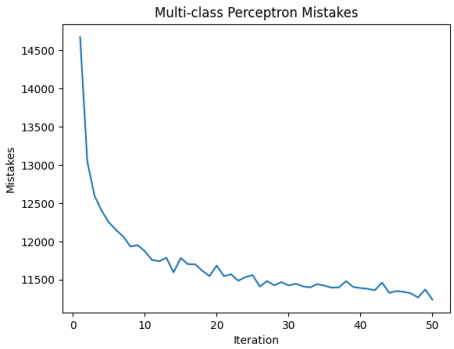




**d**

These are graphs for training on increments of 100. We can see that as increments become bigger, accuracy becomes higher and more consistent. Also these two graphs look very similar to each other.

## 2

**a**




This is as far as I can get because my computer doesn't have enough time to run through the data. The implementations are complete, but no graph are present. More information are in README.