HW 1

11581172 - John Ye

September 11, 2024

1

1.1

1.1a

No, the decision boundary of the voted perceptron may be non-linear. While each perceptron will have a linear decision boundary, but with the voted perceptron, there are multiple decision boundaries, and the combination of multiple linear decision boundaries could be non-linear.

1.1b

Yes, the decision boundary of the averaged perceptron is linear. Since the averaged perceptron takes the average of all other weight vectors and generate one decision boundary based on that averaged weight vector, it only has one decision boundary. Therefore, the decision boundary of the averaged perceptron is linear.

1.2

We first set the w_{t+1} to be the solution of the optimization problem where

$$\begin{aligned} w_{t+1} &= \min_{w} \frac{1}{2}\|w - w_t\|^2 \\ s.t. & y_t(w \cdot x_t) \geq 1 \end{aligned} \tag{1}$$

Then we find the lagrangian

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

Next, we take the optimize for w

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

by setting the derivative to zero, we can get

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

Substitute w back to the lagrangian, we have

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

Since we have a margin M, we let $L(w,t) \geq M$ and solve for τ

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

Then, the weight update for achieving margin M would be

$$\begin{aligned} w_{t+1} &= w_t + \tau y_t x_t \\ &= w_t + \frac{M - y_t(w_t \cdot x_t)}{\|x_t\|^2} y_t x_t \end{aligned} \tag{2}$$

1.3

1.3a

For each weight update, instead of the regular update

$$w_{i+1} = w_i + \tau y_i x_i$$

we add the importance weight h_i in to the function and use the following weight update instead

$$w_{i+1} = w_i + \tau h_i y_i x_i$$

Therefore, the higher the importance weight is, the larger the weight would be updated, which affect the model more than lower importance weight.

1.3b

One way to solve the given problem is to for the data with an importance weight of h_i, we dupilicate the corresponding data for i times and store it. Then, instead of update weight for each data, we can simply count the how many time each data point appears. By using this method, the computational cost can be reduced significantly. However, the memory usage is increased since we need to store more data than before.

1.4

1.4a

For the given problem, since we only interested in the accuracy of positive examples, we could assign a very large learning rate to the update for positive examples, and a very small learning rate to the update for the negative examples. Then, when the weight is updating, the positive examples will affect more on the weight than the negative ones. The weight update should look like the following
For positive examples

$$w_{i+1} = w_i + \tau_p y_i x_i, where\ \tau_p \approx 1$$

For negative examples

$$w_{i+1} = w_i + \tau_n y_i x_i, where\ \tau_n \approx 0$$

1.4b

One way to solve the given problem is to make sure the percetpron is trained with equal amount of positive and negative data examples. For each iteration, we can randomly select the same amount of positive and negative examples to create a balanced training set. Since we are only interested in the positive examples, by doing this, the overall negative examples used for the perceptron is decreasing, the algorithm will focus more on the positive examples.

2

2.1

2.1a

Here is the graph for number of mistakes with iterations for Binary Classification for Perceptron and PA algorithm.

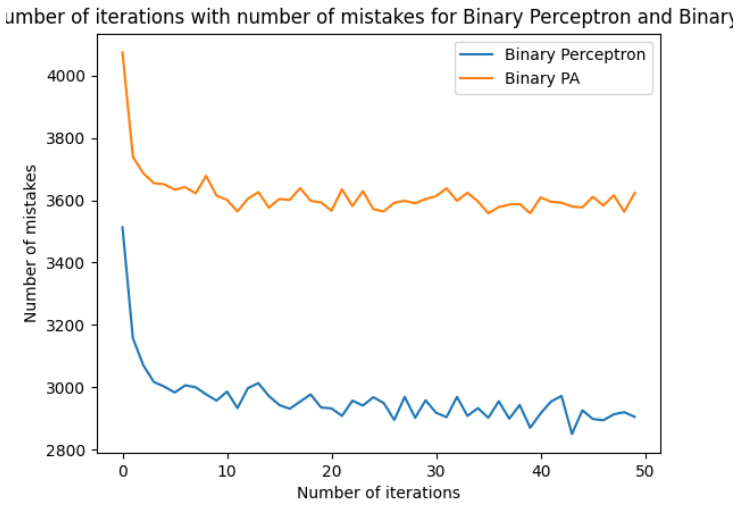

Figure 1: Number of Mistake with Iterations

From the graph, we can see that the Binary PA has more errors than Binary Perceptron. Both Perceptron and PA algorithm start with a lot mistakes, but the number of mistakes reduced significantly for the first few iterations, and they keep decreasing for each iteration.

2.1b

Here is the graph for the accuracy with interations for Binary Classification for Perceptron and PA algorithm for 20 iterations.

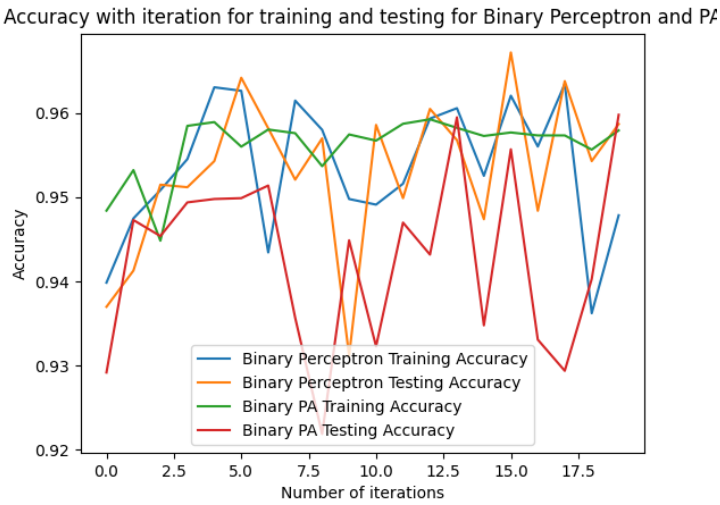

Figure 2: Accuracy with Iterations

From the graph, we can see that both Perceptron and PA algorithm have smaller accuracy at first, with the number of iterations increasing, eventually, the accuracy is also increasing. The overall accuract for the PA algorithm is higher than the Perceptron algorithm.

2.1c

Here is the graph for the accuracy with interations for Binary Classification for Perceptron and Averaged Perceptron.

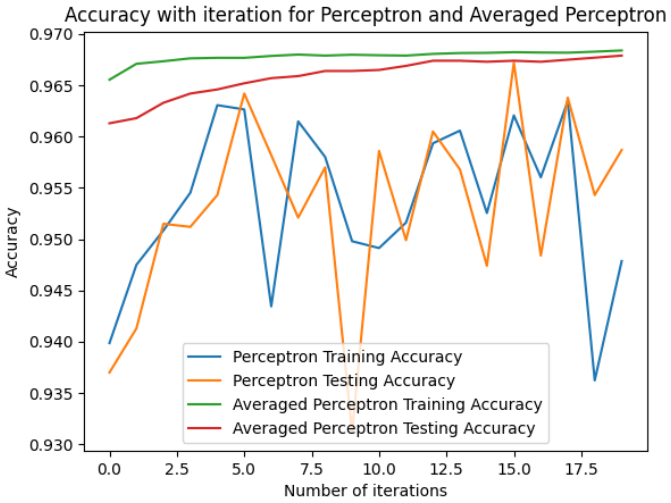

Figure 3: Accuracy with Iterations

From the graph, we can see that the accuracy for the averaged perceptron is higher that the plain perceptron. Unlike the accuracy plain perceptron , the accuracy for the averaged perceptron does not change significantly, instead, it is smoothly increasing with each iteration.

2.1d

I did not implement the learning curve for this part.

2.2

The Multi-class PA algorithm I had are makeing the same large amount of mistake with a 10% accuracy, so I will only list my observation on the Perceptron algorithm.

2.2a

Here is the graph for the number of mistakes with interations for Multi-Class Classification for Perceptron.

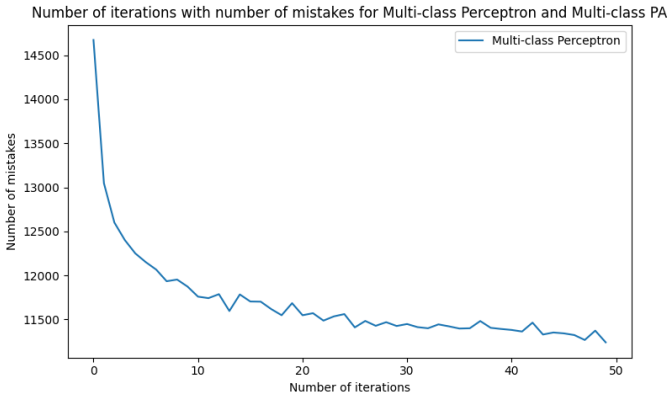

Figure 4: Number of Mistake with Iterations For Multi-Class

From the graph, we can see that the Multi-class Perceptron makes a lot of mistakes at first. The number of mistakes is reduced significantly for the first few iterations, and it makes less and less mistake while the iteration is increasing.

2.2b

Here is the graph for the number of mistakes with interations for Multi-Class Classification for Perceptron for 20 interations.

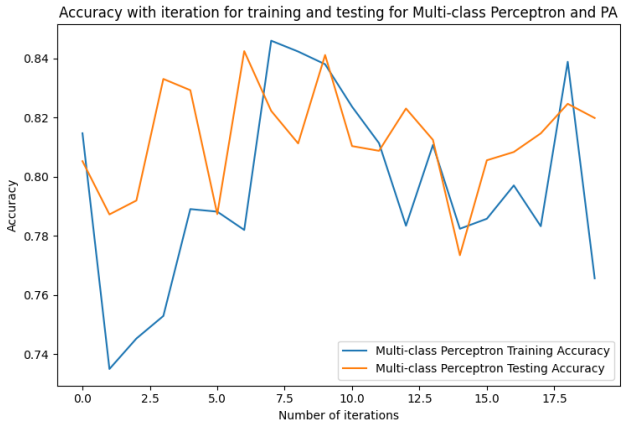

Figure 5: Accuracy with Iterations For Multi-Class

From the graph, we can see that the testing accuracy eventually is increasing but the traning accuracy is decreasing. The pattern is similiar to the Binary

Perceptron's traning and testing accuracy.

2.2c

I have the code for the averaged multi-class perceptron, but code for ploting the graph does not excute.

2.2d

I did not implement the learning curve for this part.