CPT_S 570: Homework #1

Instructor: Jana Doppa

Sheng Guan

Problem 1

Answer:
a. No. For a voted perceptron, we have multiple weight vectors.

$$f_{voted}(x) = sign(\sum_{i=1}^{m} c_i sign(< w_i, x >))$$

We store all the classifiers and each classifier will define a linear decision boundary. The weighted vectors will combine together to distinguish positive and negative examples. In short, the final decision boundary is the combination of several linear decision boundaries. For instance, vector(0,1) and (1,0) will define a upper right quadrant and instance will only be labeled as positive if it locates in this upper right quadrant region, and this is not linear.

b. Yes. For an averaged perceptron, we will average the weight vectors

$$f_{average}(x) = sign(\sum_{i=1}^{m}(< c_i w_i, x >))$$

After the average, we will get a single weight vector that defines a linear decision boundary.

Problem 2

Answer:
With margin M, the optimization problem becomes:

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

The Lagrangian:

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

Solve for the dual: $\max_{\tau \geq 0} \min_{w} \mathcal{L}(w, \tau)$

Optimize for w: $\frac{\partial \mathcal{L}}{\partial w} = w - w_t - \tau y_t x_t$

Set the derivative to zero: $w = w_t + \tau y_t x_t$

Substitute back into the Lagrangian:

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

Dual optimization problem

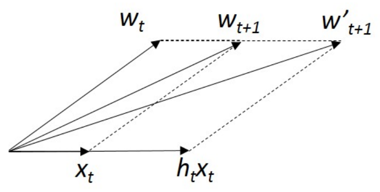
$$\max_{\tau \geq 0} -\frac{1}{2}||x_t||^2\tau^2 + \tau(M - y_t(w \bullet x_t))$$

Solve it:

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

Problem 3

Answer:
For the plain perceptron, when the mistake happens, the update will take $w_{t+1} to w_t + y_t x_t$, that treats every instance equally. Here we will take one step further, for instance with larger importance weight, we will update the weights with larger steps. Correspondingly, for instance with smaller importance weight, we update the weights more conservatively.
So we can get a new weight update equation as : $w_{t+1} \leftarrow w_t + h_t y_t x_t$.
To justify the correctness of the new update equation, here we consider an example where $h_i > 1$. Consider the plain perceptron, when we make an mistake and y_t now is 1, we want to move the vector closer to x_t (here we move tow_{t+1} according to the plain perceptron), as shown below

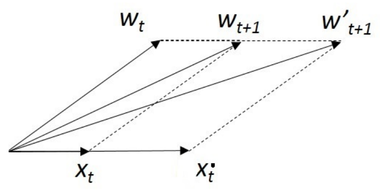


when $h_i > 1$ we know that this example is more important than $1 \bullet x_t$ so we want to move the updated weight vector closer to x_t(here move to w'_{t+1}). This also applies to when y_t is -1 where we want to move the vector far from x_t. Thus here we can use $w_{t+1} \leftarrow w_t + h_t y_t x_t$ to update the vector.

b. By looking at the update function

$$w_{t+1} \leftarrow w_t + h_t y_t x_t$$

We can modify the input data to achieve the same effect. We can treat the $h_t x_t$ as x'_t and feed x'_t as the input for the plain perceptron. Same as the above example, if we feed the x'_t as input, we still can get w'_{t+1} as the output.



Problem 4

Answer:

a. We can consider to give an additional importance weight to the training examples as Problem 3 indicates. We can add a parameter $\alpha < 1$ to the algorithm. For $y_t = -1$ use $w_{t+1} \leftarrow w_t + \alpha y_t x_t$ to update the vector when meeting a mistake. For $y_t = 1$ we still use $w_{t+1} \leftarrow w_t + y_t x_t$ when meeting a mistake. Since the negative instance : the positive instance = 9 :1, we can let α here as 1/9. We make conservative weight update when a negative instance makes an influence. However, we make significant weight update when a positive instance makes an effect. In this way, we make our weight vector more influenced by the positive instances, hopefully we can achieve a higher accuracy of positive examples.

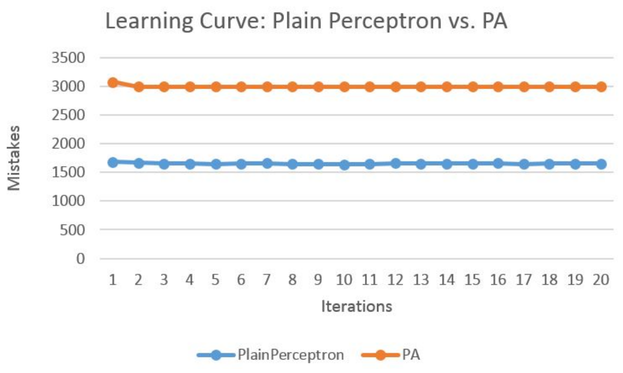
b. We can treat above mentioned α as a frequency ratio. Basically, we would like to use oversampling method to balance the training example.

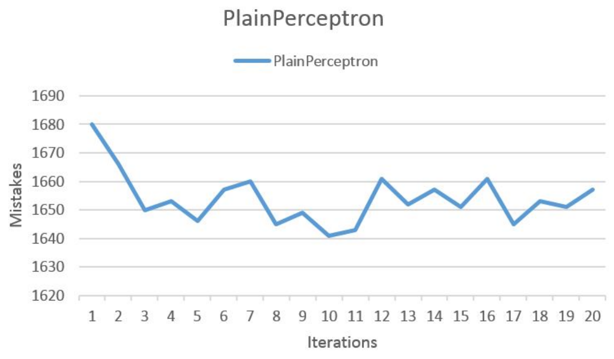
We duplicate all positive examples 9 times and then shuffle the whole data. The idea behind is still to make the vector closer to the positive one. At the same time we would like to keep the information of negative examples as much as possible, That's the reason we don't down-sample the negative instances to get a new training dataset.

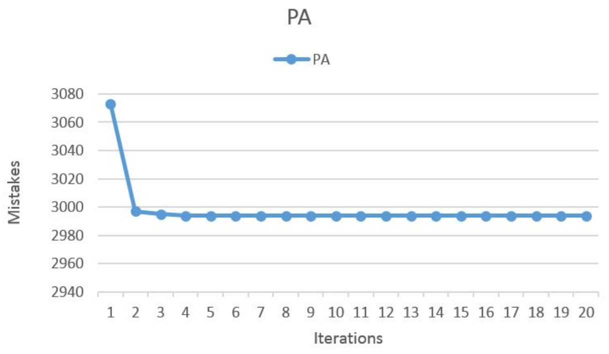
Problem 5

Answer for 5.1:

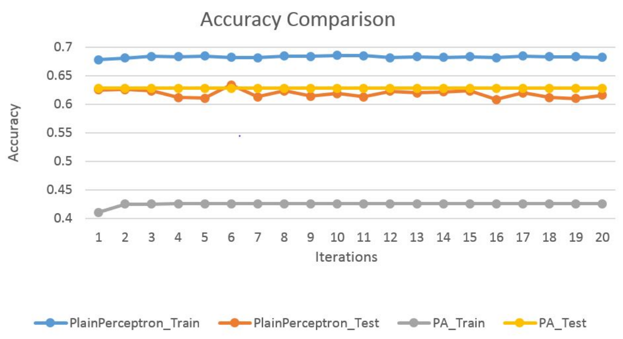
a. To further analyze the convergence, I provide two more pictures of plain Perceptron and PA algorithm separately. We can see that the learning curve for plain Perceptron algorithm cannot converge at iteration 20. One possible reason is that the dataset itself is not linearly separable. Another possible reason is learning rate 1 is too large for this dataset. However, the PA glgorithm can converge quickly.



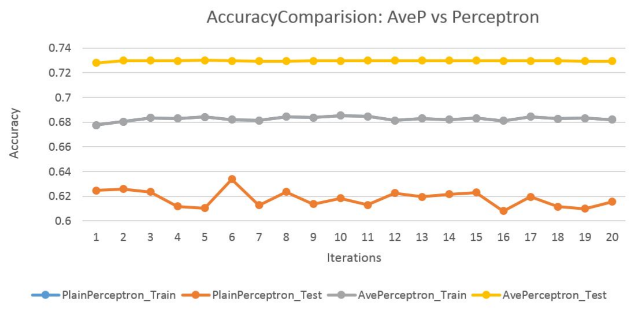




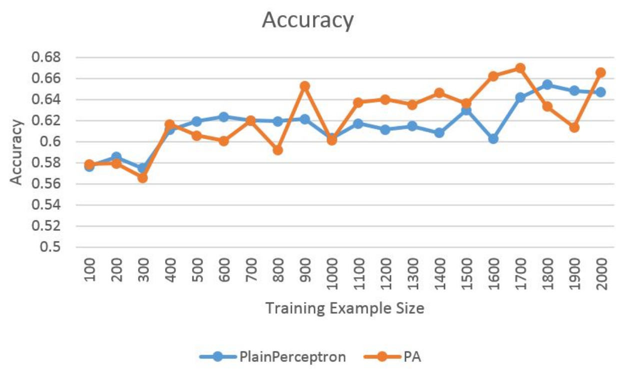
b.The below figure shows the accuracy curve of both plain Perceptron and PA for both training data and testing data. The accuracy of testing on both plain Perceptron algorithm and PA algorithm for this dataset is very close. More precisely, PA algorithm is slightly better. For the plain Perceptron algorithm, the accuracy of the training data is better than the accuracy of the testing data. It makes sense since usually, the training performance is better. However, for PA algorithm, since the margin constraints must be considered during the training process for each training example, it trades the training accuracy for the higher confidence of classification. Hence, the accuracy of training in PA algorithm is the lowest.



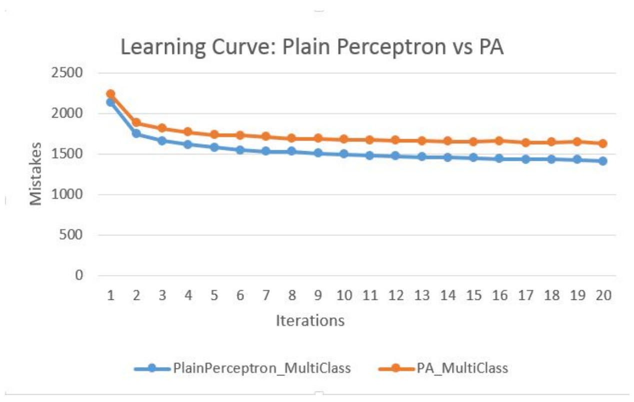
c. The result is shown as follows. The accuracy of averaged perceptron is better than plain perceptron on testing data since it is insensitive to the order of examples. The training accuracy of averaged perceptron and plain perceptron is coincide since our binary classification is an evolution of multi-class problem.

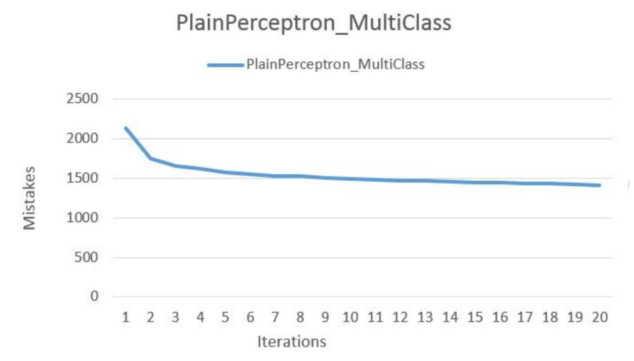


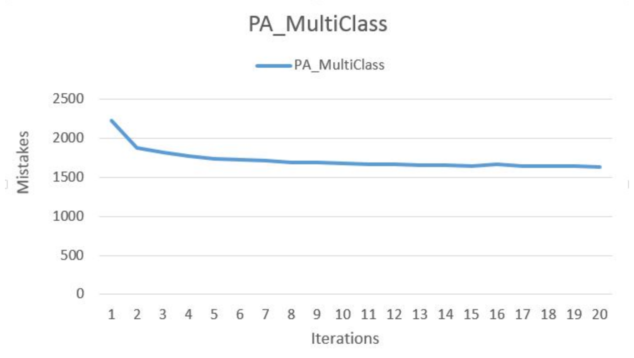
d. The result is shown as follows. We can see that with the increase of the number of training example, the accuracy does not perform monotonously increase. One possible reason is that some data in this range is not linearly separable. However, the basic trend is to increase the accuracy, which indicates the more training data is helpful to get better accuracy performance.



Answer for 5.2:
a.To further analyze the convergence, I provide two more pictures of plain Perceptron and PA algorithm separately. We can see that the learning curve for plain Perceptron algorithm and PA algorithm both have the convergence trend.

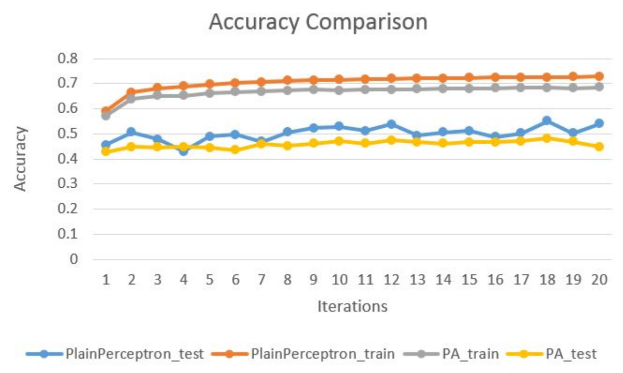




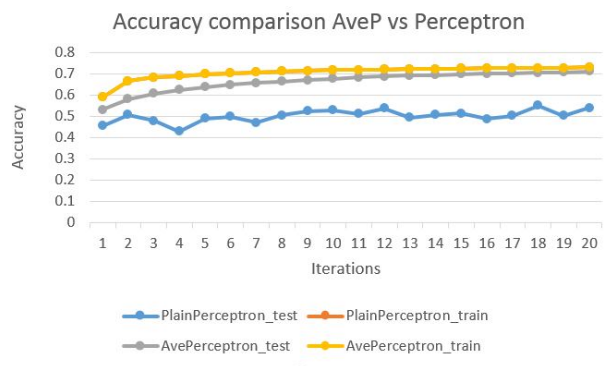


b.The below figure shows the accuracy curve of both plain Perceptron and PA for both training data and testing data. The accuracy of testing performance of plain Perceptron is better than PA. For the plain Perceptron algorithm, the accuracy of the training data is better than the accuracy of the testing data. It makes sense since usually, the training performance is better. However, for PA algorithm,I get a high training accuracy surprisingly. I personally think I didn't select the correct loss function here when updating the weight in Representation II form. I tried to use (0,1) loss and hinge loss. However, (0,1) loss gives me very high training accuracy. However, hinge loss gives me very low testing accuracy. As a tradeoff, I use the (0,1) loss here.

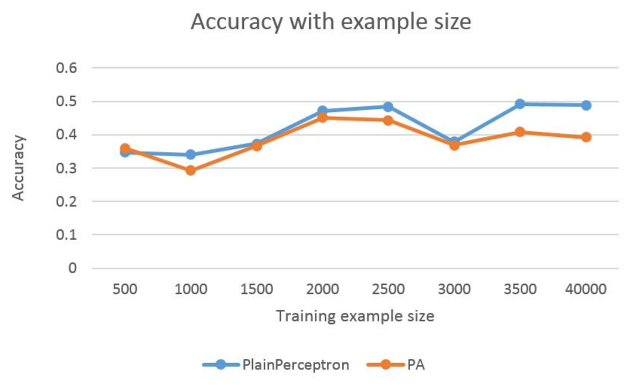
$$\begin{aligned} \mathbf{w}_{t+1} &= \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \\ \text{s.t.} & \quad \mathbf{w}\cdot F(\mathbf{x}_t, y_t) - \mathbf{w}\cdot F(\mathbf{x}_t, \hat{y}_t) \geq \ell(\hat{y}_t, y_t) \end{aligned}$$



c. The result is shown as follows. The accuracy of averaged perceptron is better than plain perceptron on testing data since it is insensitive to the order of examples. The training accuracy of averaged perceptron and plain perceptron is coincide since our binary classification is an evolution of multi-class problem.



d. The result is shown as follows. We can see that with the increase of the number of training example, the accuracy does not perform monotonously increase. One possible reason is that some data in this range is not linearly separable.


Accuracy with example size
Accuracy
Training example size
PlainPerceptron
PA