Oregon State University

AI_534_001_F2021 MACHINE LEARNING

Implementation Assignment 3

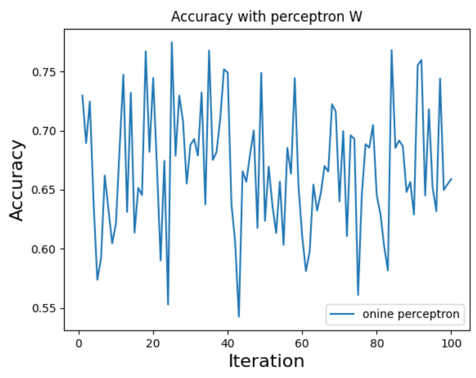
Professor: Xiaoli Fern
Student: Chengxu Xu

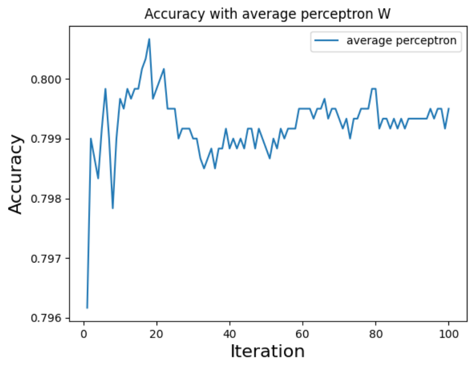
Part1

(a)

Train data:

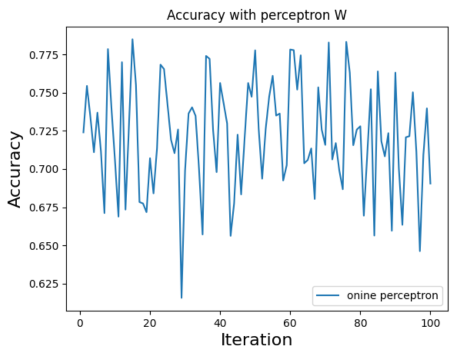
online perceptron average perceptron

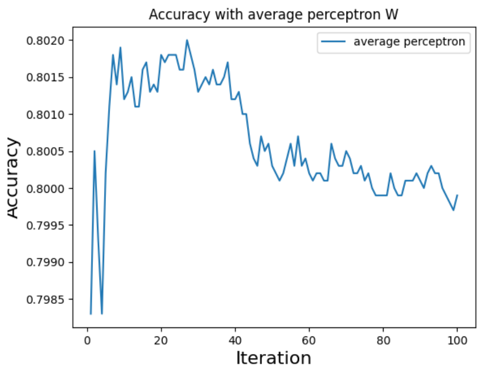




Validate data:

online perceptron average perceptron



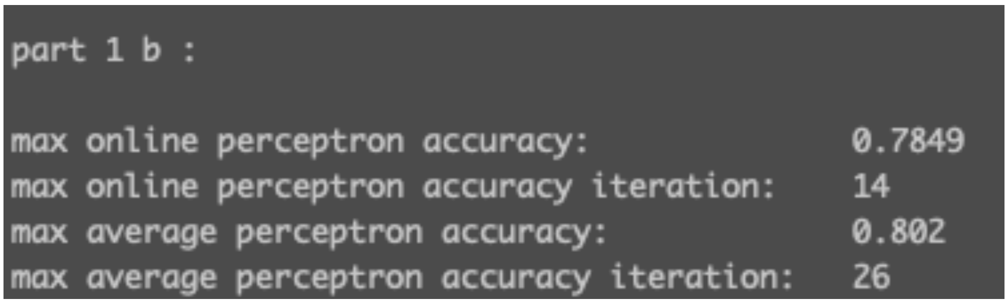


After observing the generated training/validation accuracy curves I found that the accuracy of the online perceptron fluctuates a lot for both training and validation data. Whereas the average perceptron accuracy variation decreases gradually, reaching the highest accuracy in 24 iterations, and the accuracy tends to decrease as the number of iterations increases.

I think one possible reason for this phenomenon is that the average perceptron averages W in the algorithm, so as the number of iterations increases, the impact of a single result on the final W becomes smaller and smaller, so the fluctuation of the curve gradually decreases. In contrast, the online perceptron iterations do not affect each other, we only look at one example at a time, so the accuracy fluctuates irregularly.

(b)

Consider the best stopping point is the one that stops when the accuracy reaches its maximum. The maximum accuracies for online perceptron and average perceptron and their corresponding iteration are shown below.



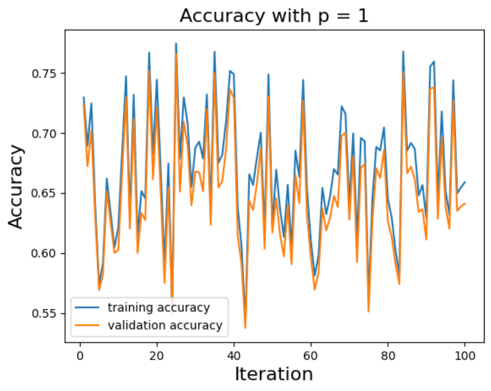
```
part 1 b :

max online perceptron accuracy:             0.7849
max online perceptron accuracy iteration:   14
max average perceptron accuracy:            0.802
max average perceptron accuracy iteration:  26
```

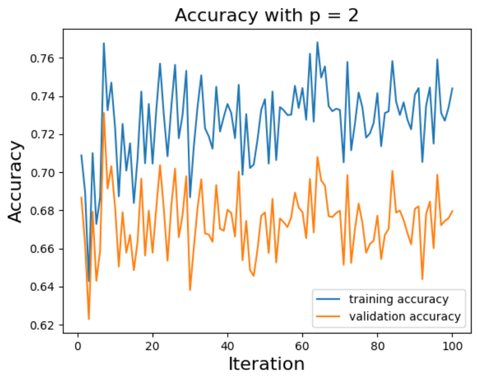
According to the curve analysis, the online perceptron is more sensitive and different stopping points may lead to very different accuracies.

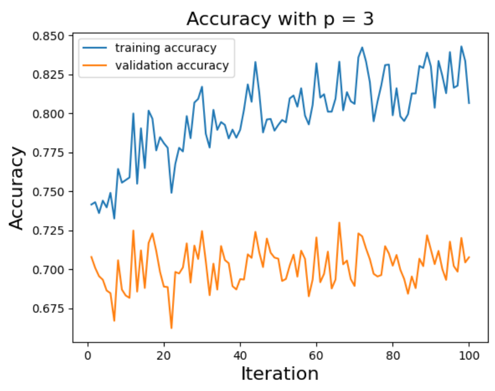
This sensitivity has many practical applications, such as when the order of the training examples is important and we can use online perceptron to better distinguish the order, or when the data is not linearly differentiable, which means that convergence is not guaranteed, when online perceptron would be very practical.

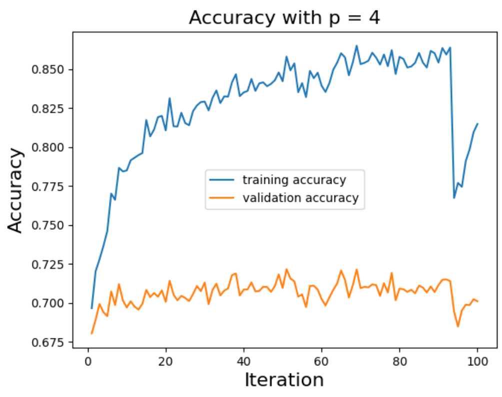
Part2a

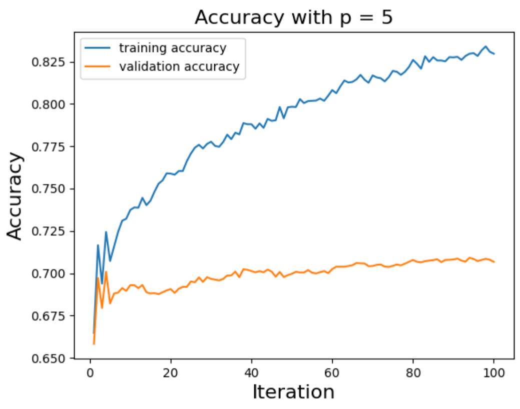
(a)







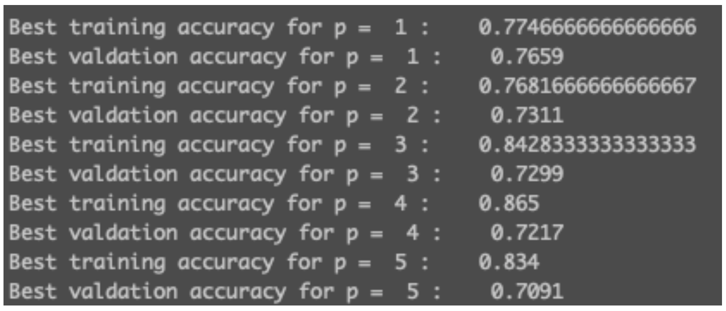




By observing the generated curves, I found that the overlap between the training and validation curves decreases as p increases, with the highest overlap at p=1. At the same time, the fluctuation of the curve also decreases as p increases.

I think one of the possible reasons for this phenomenon is that the training data is too small, and the number of iterations is too large and the lack of regularization of the kernelized perceptron leads to the overfitting phenomenon.

(b)

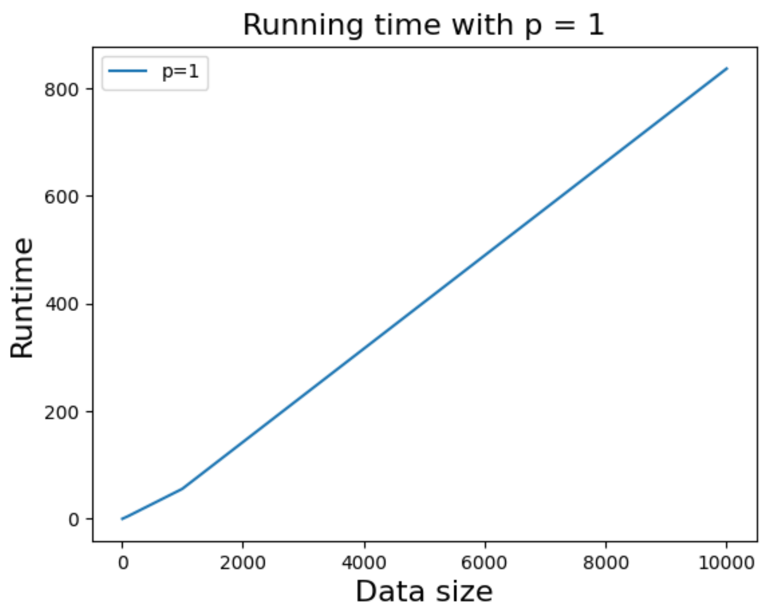


```
Best training accuracy for p =  1 :   0.7746666666666666
Best valdation accuracy for p =  1 :   0.7659
Best training accuracy for p =  2 :   0.7681666666666667
Best valdation accuracy for p =  2 :   0.7311
Best training accuracy for p =  3 :   0.8428333333333333
Best valdation accuracy for p =  3 :   0.7299
Best training accuracy for p =  4 :   0.865
Best valdation accuracy for p =  4 :   0.7217
Best training accuracy for p =  5 :   0.834
Best valdation accuracy for p =  5 :   0.7091
```

The best validation accuracy is achieved with p=1. I think p changes the accuracy by changing the amount of data generated by the polynomial kernel, and as p increases, the overfitting gets worse.

(c)

Empirical runtime vs training data size when iteration = 100

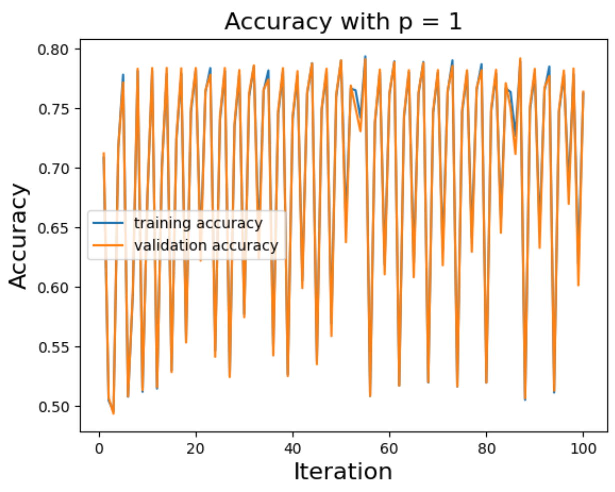


The asymptotic time complexity of my algorithm is O(n^2). Based on the generated curves, the empirical runtime matches up with the asymptotic analysis.

Part2b

(a)

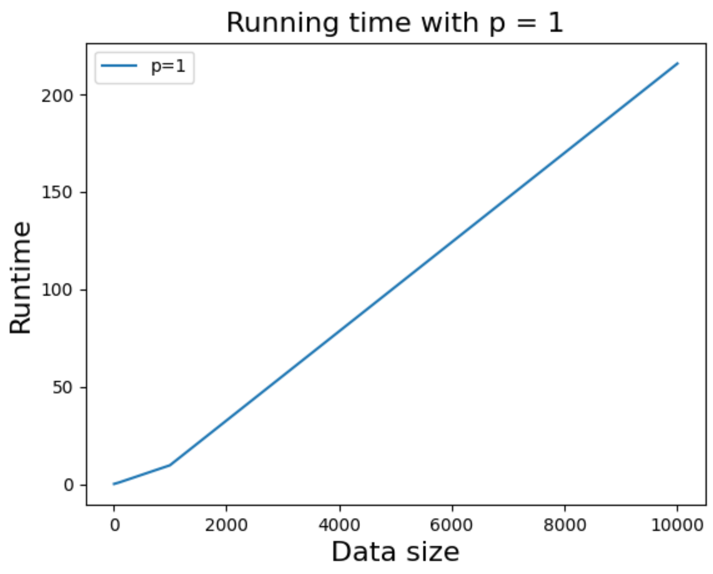
According to the data analysis of part2a, the best accuracy occurs when p=1, so the following figure is the image of the training data and validation data of the batch kernel perceptron achieved when p takes the value of 1 against the number of iterations.



Compared to Part 2a, under the same p value of 1, the batch algorithm shows sharp fluctuations in accuracy and does not show a trend of convergence. However, the overlap between the accuracies generated for the training and validation data is very high, and I think one possible explanation is that each iteration creates a new batch, thus reducing the impact from iteration to iteration.

Using different learning rates will result in different predictions x, and therefore will have a different impact on the outcome of the predictions and the accuracy of the algorithm generated. The larger the learning rate, the more unstable the weight changes are.

(b)



Psudocode:

```
while iteration < ba_max_iteration:
       batch = np.zeros(len(Y))
       for i in range(N):
           u = np.sign(np.dot(K[i], np.multiply(alpha,
Y)))
           if Y[i] * u <= 0:
               batch[i] += 1

       alpha += batch

       pred = prediction(alpha, kernel_function(X2, X,
p), Y)
       if Y2 is not None:
           acc.append(get_accuracy(pred, Y2))

       iteration += 1
    return alpha, acc,
```

The asymptotic runtime of the batch kernel perceptron as a function of the number of training examples n is O(n^2). The empirical runtime matches up with the asymptotic analysis.

After observing and comparing the curves generated by part2ac with those generated by paer2bc, I found that there is a significant difference in the running time between the two algorithms, with the batch algorithm running longer for the same p values. I think one possible reason for this is that the batch algorithm uses a new batch to store the alpha in each iteration, so the running time is reduced