CS534 HW 3 report

Student ID: 933-953-476

Name: Dayeon Oh

Part1.

(a) Apply the implemented algorithm to learn from the training data with *maxitr* = 100.

```
Accuracy of training data with online perceptron : 72.50746268656715 with iteration : 10
Accuracy of training data with average perceptron : 78.97512437810946 with iteration : 10
Accuracy of validation data with online perceptron : 72.52525252525253 with iteration : 10
Accuracy of validation data with average perceptron : 79.05050505050505 with iteration : 10

Accuracy of training data with online perceptron : 74.29850746268657 with iteration : 20
Accuracy of training data with average perceptron : 79.02487562189054 with iteration : 20
Accuracy of validation data with online perceptron : 74.4040404040404 with iteration : 20
Accuracy of validation data with average perceptron : 79.0909090909091 with iteration : 20

Accuracy of training data with online perceptron : 77.88059701492537 with iteration : 30
Accuracy of training data with average perceptron : 79.04477611940298 with iteration : 30
Accuracy of validation data with online perceptron : 78.72727272727272 with iteration : 30
Accuracy of validation data with average perceptron : 79.11111111111111 with iteration : 30

Accuracy of training data with online perceptron : 73.5223880597015 with iteration : 40
Accuracy of training data with average perceptron : 79.0547263681592 with iteration : 40
Accuracy of validation data with online perceptron : 73.27272727272728 with iteration : 40
Accuracy of validation data with average perceptron : 79.0909090909091 with iteration : 40

Accuracy of training data with online perceptron : 71.93034825870647 with iteration : 50
Accuracy of training data with average perceptron : 79.08457711442786 with iteration : 50
Accuracy of validation data with online perceptron : 71.65656565656565 with iteration : 50
Accuracy of validation data with average perceptron : 79.13131313131314 with iteration : 50
```

```
Accuracy of training data with online perceptron : 76.76616915422886 with iteration : 60
Accuracy of training data with average perceptron : 79.06467661691542 with iteration : 60
Accuracy of validation data with online perceptron : 77.11111111111111 with iteration : 60
Accuracy of validation data with average perceptron : 79.0909090909091 with iteration : 60

Accuracy of training data with online perceptron : 73.45273631840796 with iteration : 70
Accuracy of training data with average perceptron : 79.06467661691542 with iteration : 70
Accuracy of validation data with online perceptron : 73.25252525252526 with iteration : 70
Accuracy of validation data with average perceptron : 79.07070707070707 with iteration : 70

Accuracy of training data with online perceptron : 75.71144278606965 with iteration : 80
Accuracy of training data with average perceptron : 79.06467661691542 with iteration : 80
Accuracy of validation data with online perceptron : 75.91919191919192 with iteration : 80
Accuracy of validation data with average perceptron : 79.07070707070707 with iteration : 80

Accuracy of training data with online perceptron : 76.57711442786069 with iteration : 90
Accuracy of training data with average perceptron : 79.07462686567163 with iteration : 90
Accuracy of validation data with online perceptron : 76.94949494949495 with iteration : 90
Accuracy of validation data with average perceptron : 79.0909090909091 with iteration : 90

Accuracy of training data with online perceptron : 73.92039800995025 with iteration : 100
Accuracy of training data with average perceptron : 79.08457711442786 with iteration : 100
Accuracy of validation data with online perceptron : 73.75757575757575 with iteration : 100
Accuracy of validation data with average perceptron : 79.11111111111111 with iteration : 100
```

Figure 1. Average Perception implementation

In figure 1, it shows the training accuracy and validation accuracy of both online perceptron and average perceptron result according to the iteration time. The maximum value of iteration is 100, so I printed out the 10^{th}, 20^{th}, ... ,100^{th} iteration results.

Plot the train and validation accuracy of online perceptron and average perceptron.

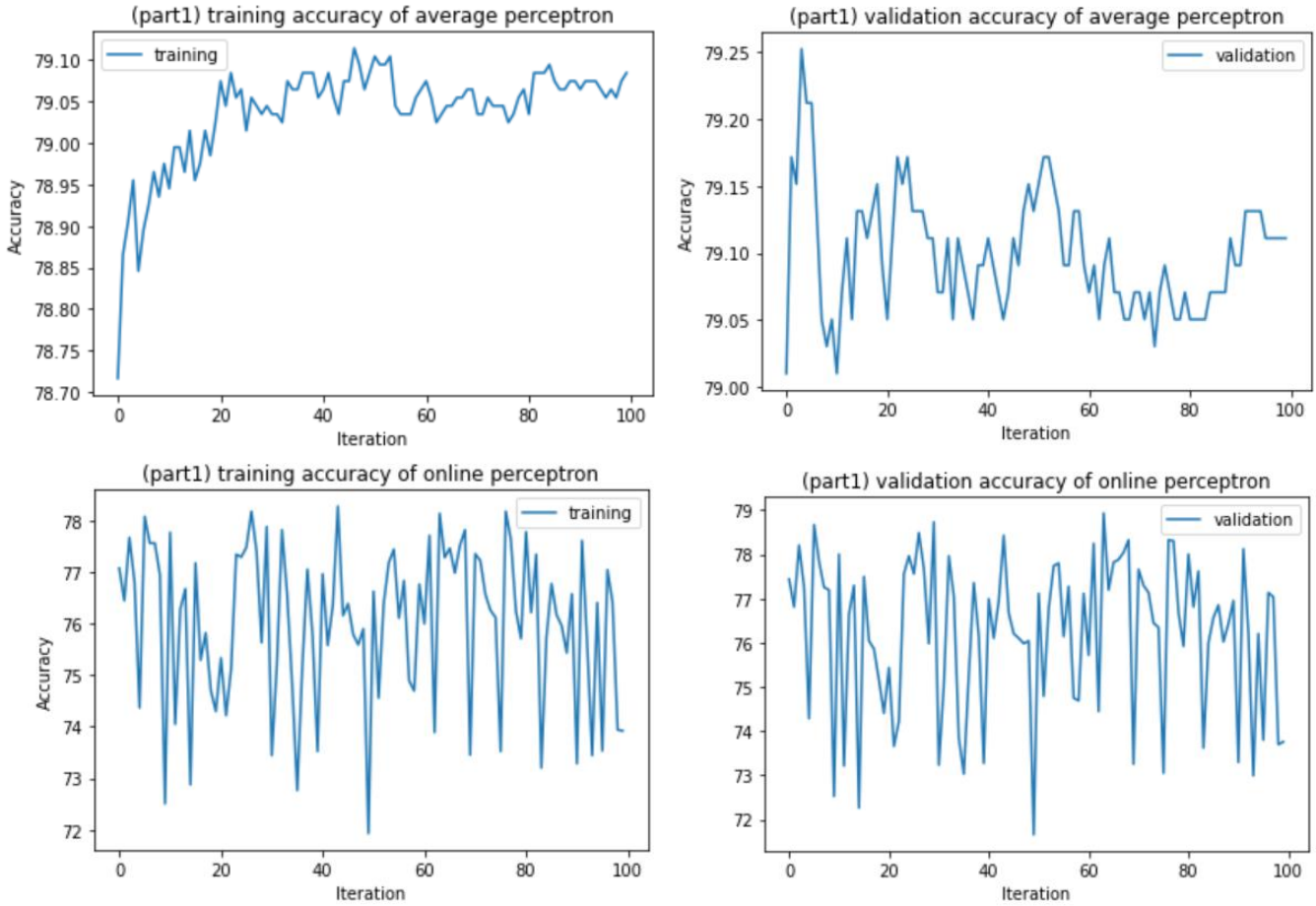

Figure 2. Training and Validation Accuracy of Online and Average Perceptron

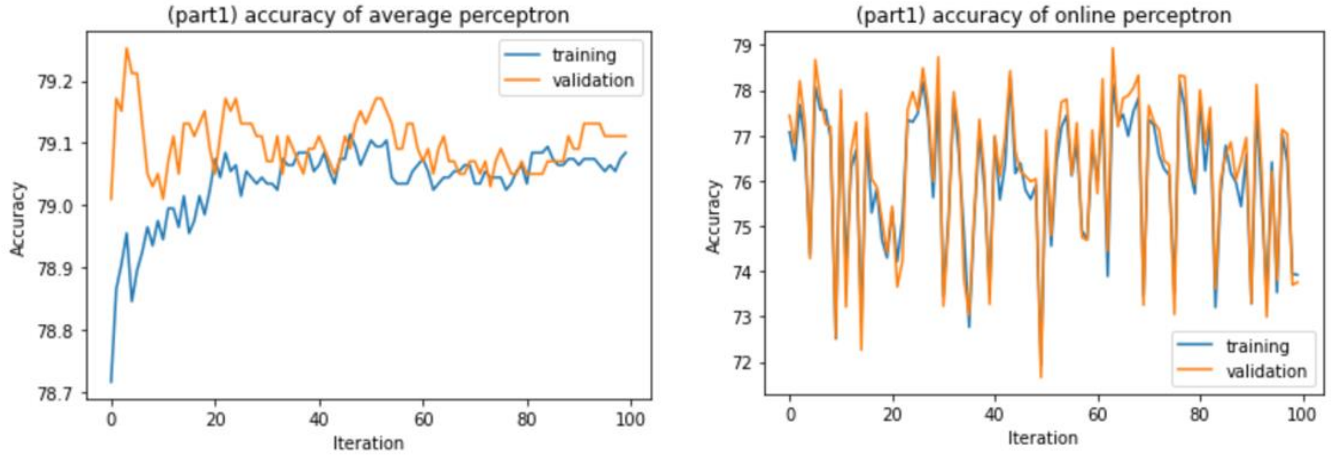

Figure 3. Accuracy of average and online perceptron

(b)

1) What are the observations when comparing the training accuracy and validation accuracy curves of the average perceptron with those of the online perceptron?

In the figure2, top 2 figures are each training accuracy and validation accuracy of online perceptron and bottom 2 figures are accuracy results of average perceptron. In the figure3, I plotted the train dataset accuracy and validation dataset accuracy of both online and average. We can see that result of online perceptron both training accuracy and validation accuracy tends to diverge a lot according to the number of iteration, the average perceptron, on the other hand, tends to diverge less compared to the online perceptron.

2) What are your explanation for the observations?

Since the online perceptron updates on every error it makes, it is heavily influenced by the final weight vector obtained in its runtime. For example, let's say there is 100 dataset and first 95 dataset were classified correct but last 5 were mistake, then this online perceptron will get updated and it could change the whole next iteration prediction. So, there's a possibility that its accuracy diverges a lot since it depends huge on final weight vector. In contrast, in the case of average perceptron, it maintains the collection of weight vectors and its counter time and predict according to the average weight vector. Average perceptron always updates its weight vector regardless of whether there is a mistake in prediction or not, so it allows its weight vector not to skewed based the latest dataset it receives. Therefore, average perceptron will show more gentle divergence of accuracy compared to the online perceptron.

(c) Use the validation accuracy to decide the best number of iterations to stop.

	iteration	accuracy
0	4	79.252525
1	5	79.212121
2	6	79.212121
3	2	79.171717
4	23	79.171717
5	25	79.171717

	iteration	accuracy
0	1	79.010101
1	11	79.010101
2	9	79.030303
3	74	79.030303
4	8	79.050505
5	10	79.050505

Figure 4. Highest accuracies and Lowest accuracies

In figure 4, left shows the top 6 highest validation accuracy and right shows the lowest 6 validation accuracy. Also, if we see top-right graph in figure 2, we can see that on iteration 4 and 5 it shows the highest validation accuracy. If we closely look at the detail data in the figure 4, iteration 4 and 5 is the highest and iteration 9, 11 shows the lowest accuracy value (ignored the first iteration because it gets higher until 4). Therefore, we can say that best number of iterations to stop is 4. The validation accuracy does not gradually increase because of the over-fitting, so we can say that large iteration does not guarantee the best validation accuracy.

Part 2a.

(a) Apply the kernalized perceptron with different p values in [1, 2, 3, 4, 5] with *maxiter* = 100.

```
Train accuracy with p value 1 is : 74.80597014925372 itr_cnt : 100
Validation accuracy with p value 1 is : 74.74747474747475 itr_cnt : 100
Train accuracy with p value 2 is : 78.79601990049751 itr_cnt : 100
Validation accuracy with p value 2 is : 76.1010101010101 itr_cnt : 100
Train accuracy with p value 3 is : 80.25870646766168 itr_cnt : 100
Validation accuracy with p value 3 is : 75.43434343434343 itr_cnt : 100
Train accuracy with p value 4 is : 81.78109452736318 itr_cnt : 100
Validation accuracy with p value 4 is : 75.79797979797979 itr_cnt : 100
Train accuracy with p value 5 is : 82.35820895522387 itr_cnt : 100
Validation accuracy with p value 5 is : 74.62626262626263 itr_cnt : 100
```

Figure 5. Online kernelized perceptron implementation

```
Accuracy of training data with online perceptron : 73.92039800995025 with iteration : 100
Accuracy of training data with average perceptron : 79.08457711442786 with iteration : 100
Accuracy of validation data with online perceptron : 73.75757575757575 with iteration : 100
Accuracy of validation data with average perceptron : 79.11111111111111 with iteration : 100
```

Figure 6. vanilla online perceptron result

Figure 5 is the result of 100th iteration of train accuracy and validation accuracy according to the p values. When the p value is 1, it will return to the vanilla online perceptron algorithm which we did in part1. So, figure 6 is the 10th iteration of online perceptron and average perceptron. If we compare online perceptron value, training with 73.92 and validation with 73.75, we can see that it has almost similar result with first two accuracy in figure 5.

(b) Record and plot the train and validation accuracy as a function of training iterations.

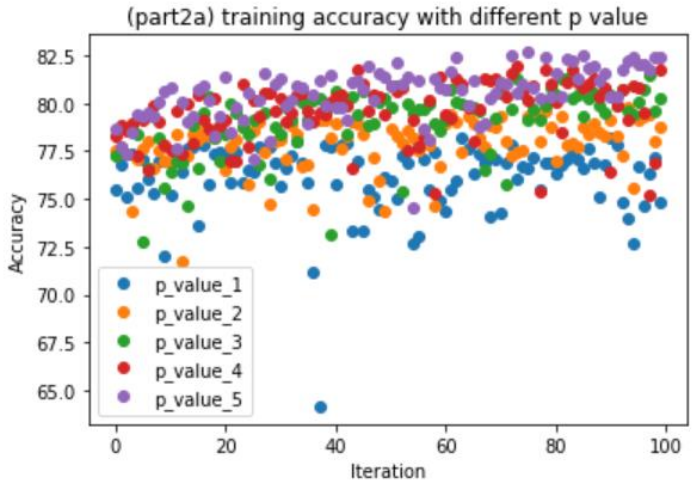

Figure 7. train accuracy according to p values

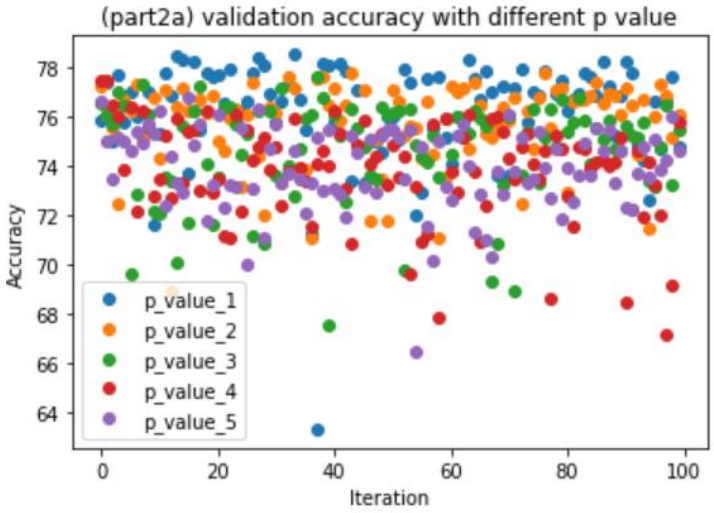

Figure 8. validation accuracy according to p values

A Figure 7 is the result of 5 different training accuracy result according to the p values. We can see the purple dots are generally at the highest section in the graph, which means p value with 5. A Figure 8 is the result of 5 different validation accuracy result according to the p values. In this graph unlike figure 8, blue dots are generally at the highest section of the graph, where p value is 1.

(c)

1) Plot the recorded best validation accuracy versus p.

P value	Best Accuracy	Lowest Accuracy	Accuracy Mean	Accuracy Std
P = 1	78.5253	63.3333	76.2226	2.0770
P = 2	77.7374	68.9697	75.6715	1.7490
P = 3	77.5960	67.5556	74.6553	2.0328
P = 4	77.4747	67.1920	73.8378	2.0142
P = 5	76.7475	66.5051	73.9141	1.6516

Table 1. Validation accuracy values

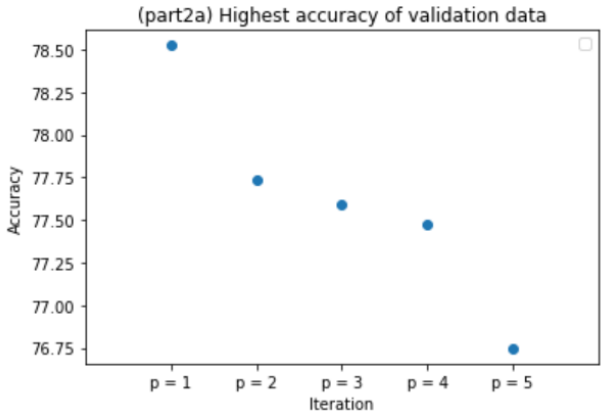

Figure 9. Highest validation accuracy according to p

P value	Best Accuracy	Lowest Accuracy	Accuracy Mean	Accuracy Std
P = 1	78.2089	64.1791	76.0924	1.8827
P = 2	79.7711	71.7711	77.8579	1.3504
P = 3	81.4328	72.7761	79.0403	1.6811
P = 4	81.9005	75.1940	79.5886	1.5384
P = 5	82.6567	74.5273	80.4713	1.4208

Table 2. Training accuracy values

2) How does p is affecting the train and validation accuracy and explanation.

In table 1 and 2 shows the accuracy value of validation and train data. When we see the train data, as the p increases, the value of highest accuracy also increases. As the p value increases, it will be more efficient to classify highly dimensional dataset. However, when we see look at table 1, validation accuracy data, it shows the highest accuracy of 78.5 when the p value is 1. I assume that this is because our data is 1, and -1 which are linearly separable, so higher polynomial kernel is not necessary for classifying our dataset. Therefore, linear decision boundary where p value is 1 classifies our validation data most correctly.

(d)

1) Asymptotic runtime of algorithm

```
while (itr_cnt < max_itr):
    itr_cnt += 1
    for i in range(N):
        val = np.multiply(alpha.T, Kernal[i])
        u = (np.dot(val,y))

        if (u*y[i] <= 0):
            alpha[i] = alpha[i] + 1
```

Figure 10. online kernelized perceptron algorithm

The online kernelized perceptron algorithm makes prediction for each training example and update when there is a mistake. So, loop through the whole range of training data, which makes $O(n)$. Then, inside the loop it does multiplying two matrices which makes $O(n^2)$. Since this complexity is inside the loop, final complexity of the algorithm is $O(n^2) * O(n) = O(n^3)$

2) empirical runtime of algorithm through iteration

```
time took : 5.374983787536621 with p value : 1 with iteration : 10
time took : 10.786147356033325 with p value : 1 with iteration : 20
time took : 16.166089296340942 with p value : 1 with iteration : 30
time took : 21.45037341117859 with p value : 1 with iteration : 40
time took : 26.778906106948853 with p value : 1 with iteration : 50
time took : 32.18042612075806 with p value : 1 with iteration : 60
time took : 37.58758091926575 with p value : 1 with iteration : 70
time took : 43.13315200805664 with p value : 1 with iteration : 80
time took : 48.09239435195923 with p value : 1 with iteration : 90
Train accuracy with p value 1 is : 74.80597014925372 itr_cnt : 100
Validation accuracy with p value 1 is : 74.74747474747475 itr_cnt : 100
time took : 53.49867033958435 with p value : 1 with iteration : 100
```

Figure 11. Empirical runtime of online kernelized perceptron algorithm

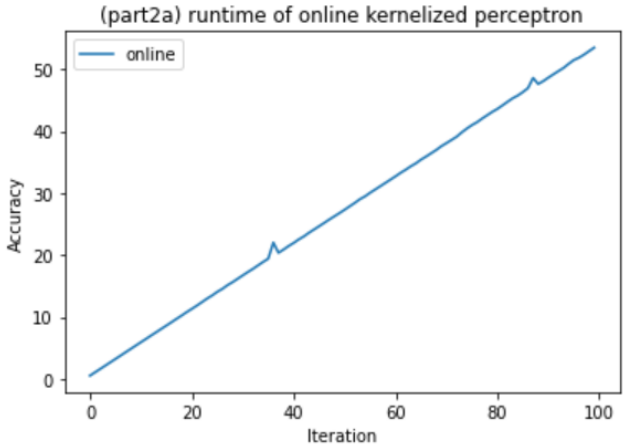

Figure 12. Empirical runtime graph of online kernelized perceptron

In figure 11 and 12, we can see the empirical runtime of online kernelized perceptron. In figure 11, I printed the runtime of 10^{th}, 20^{th}, ... ,100^{th} iteration. As the iteration gets bigger, we can see that the runtime of iteration gets longer. When we see the graph in figure 12, generally the empirical runtime gets longer as the iteration gets longer, because it goes for the while loop with the same amount of iteration time.

Part 2b.

(a) Apply the batch kernel perceptron with the best p value selected in part 2a.

```
kahng-01 ~/my_scratch/web/Codes/CS534/HW3 1095$ python3 part2b.py

Train accuracy with p value 1 is : 49.6318407960199 itr_cnt : 10
Validation accuracy with p value 1 is : 49.858585858585855 itr_cnt : 10

Train accuracy with p value 1 is : 75.75124378109453 itr_cnt : 20
Validation accuracy with p value 1 is : 75.87878787878788 itr_cnt : 20

Train accuracy with p value 1 is : 77.70149253731343 itr_cnt : 30
Validation accuracy with p value 1 is : 78.18181818181819 itr_cnt : 30

Train accuracy with p value 1 is : 75.87064676616916 itr_cnt : 40
Validation accuracy with p value 1 is : 76.1010101010101 itr_cnt : 40

Train accuracy with p value 1 is : 74.82587064676616 itr_cnt : 50
Validation accuracy with p value 1 is : 74.84848484848486 itr_cnt : 50

Train accuracy with p value 1 is : 49.62189054726368 itr_cnt : 60
Validation accuracy with p value 1 is : 49.898989898989896 itr_cnt : 60

Train accuracy with p value 1 is : 49.64179104477612 itr_cnt : 70
Validation accuracy with p value 1 is : 49.878787878787875 itr_cnt : 70

Train accuracy with p value 1 is : 49.64179104477612 itr_cnt : 80
Validation accuracy with p value 1 is : 49.838383838383834 itr_cnt : 80

Train accuracy with p value 1 is : 78.46766169154229 itr_cnt : 90
Validation accuracy with p value 1 is : 78.88888888888889 itr_cnt : 90

Train accuracy with p value 1 is : 62.71641791044777 itr_cnt : 100
Validation accuracy with p value 1 is : 61.858585858585855 itr_cnt : 100
```

Figure 13. Batched kernelized perceptron implementation

In part 2a, since we have linearly separable data $\in \{-1, 1\}$, polynomial kernel with degree 1 resulted the best accuracy, among others. So, for part 2b, I have chosen the p value 1 for the implementation and training the dataset. Above figure 13 is the result of train accuracy and validation accuracy according to the iteration count.

Changing the learning rate does not have an affect on the accuracy, this is because the result of both train accuracy and validation accuracy does not converge but diverge along the iteration value.

```
Train accuracy with p value 1 is : 62.71641791044777 itr_cnt : 100
Validation accuracy with p value 1 is : 61.858585858585855 itr_cnt : 100
Train accuracy with p value 1 is : 74.4776119402985 itr_cnt : 200
Validation accuracy with p value 1 is : 74.74747474747475 itr_cnt : 200
Train accuracy with p value 1 is : 77.55223880597015 itr_cnt : 300
Validation accuracy with p value 1 is : 77.81818181818181 itr_cnt : 300
Train accuracy with p value 1 is : 75.40298507462686 itr_cnt : 400
Validation accuracy with p value 1 is : 75.5959595959596 itr_cnt : 400
Train accuracy with p value 1 is : 77.60199004975125 itr_cnt : 500
Validation accuracy with p value 1 is : 77.87878787878788 itr_cnt : 500
Train accuracy with p value 1 is : 53.502487562189046 itr_cnt : 600
Validation accuracy with p value 1 is : 53.515151515151516 itr_cnt : 600
Train accuracy with p value 1 is : 68.40796019900498 itr_cnt : 700
Validation accuracy with p value 1 is : 67.6969696969697 itr_cnt : 700
Train accuracy with p value 1 is : 78.48756218905473 itr_cnt : 800
Validation accuracy with p value 1 is : 79.0909090909091 itr_cnt : 800
Train accuracy with p value 1 is : 75.06467661691542 itr_cnt : 900
Validation accuracy with p value 1 is : 75.2121212121212 itr_cnt : 900
Train accuracy with p value 1 is : 77.70149253731343 itr_cnt : 1000
Validation accuracy with p value 1 is : 77.93939393939394 itr_cnt : 1000
```

Figure 14. Result of 1000 iteration

Figure 14 shows the value of accuracy result with the maximum iteration 1000. We can see that it still diverges. Therefore, since the accuracy still diverges, even if we change the learning rate, the result will still diverge and show no difference in the result.

(b) Record and plot the training accuracy and validation accuracy as a function of the iterations.

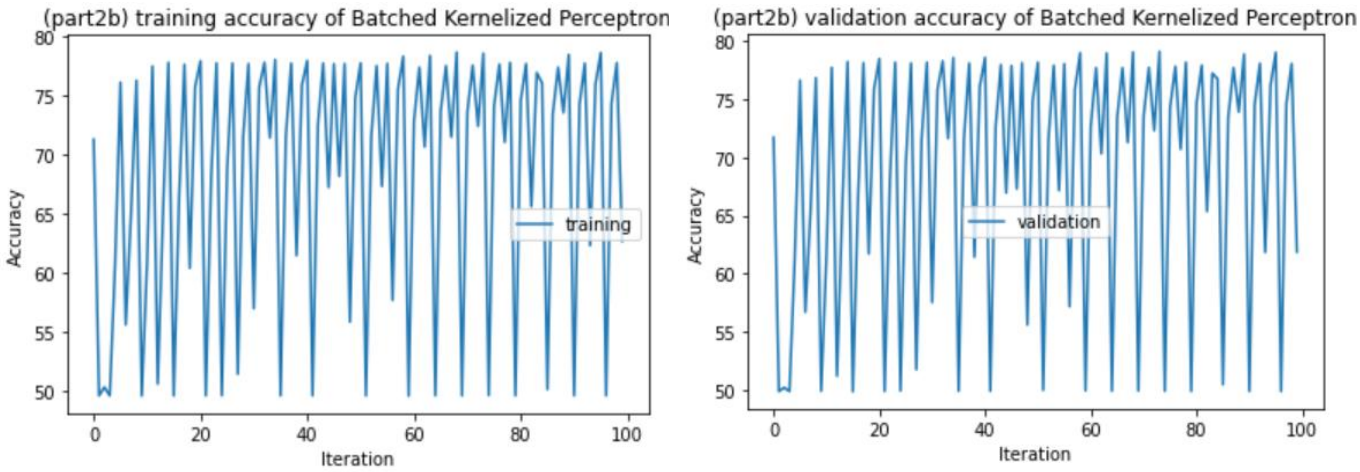

Figure 15. Training and Validation accuracy of Batched Kernelized Perceptron

As we can see in the figure 15 above, both the training accuracy and validation accuracy diverges along the number of iterations. If we look at figure 13 and 14 for accurate accuracy value of train and validation, we can also see that it diverges.

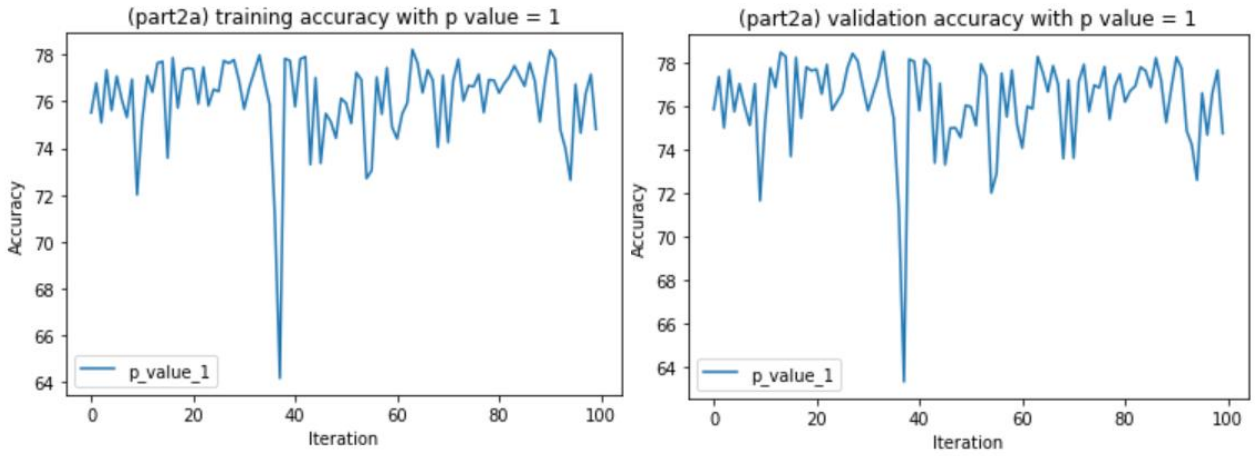

Figure 16. Part2a P = 1, accuracy results

When we compare the result of figure 15 and 16, first we can see that both figure 15 and 16 diverges. However, when doing online kernelized perceptron, the data points arrive one by one and the algorithm predicts a classification of certain example, and the algorithm decides whether to update the weight vector or not. Therefore, it updates the alpha in the size of scalar in each iteration, and we can see that it diverges from (64 ~ 78, but mostly along the value between 71 ~ 78) However, in batch kernelized perceptron, instead of looping through each separate datapoints, it calculates the whole matrices and update the alpha in the size of vector itself. So, we can see that batched kernelized perceptron diverges more bigger from (49 ~ 80) than online kernelized perceptron algorithm.

(c)

1) Asymptotic runtime of the algorithm

```
while (itr_cnt < max_itr):
    itr_cnt += 1

    val = np.multiply(alpha.T, Kernal)
    u = np.dot(val, y)
    upd = np.multiply(u, y)

    alpha = alpha + 1 * np.where(upd <= 0, 1, 0)
```

Figure 17. Batched Kernelized Perceptron Algorithm

Unlike the online kernelized perceptron algorithm, where it loops through each training dataset, batched kernelized perceptron algorithm calculate in the matrices and vector form and update the weight vector at once. So, during the iteration process, the algorithm multiply and calculate the matrices which cause the time complexity $O(n^2)$.

2) Plot the empirical runtime of the algorithm

```
kahng-01 ~/my_scratch/web/Codes/CS534/HW3 1001$ python3 part2b.py

time took : 2.578347682952881 with p value : 1 with iteration : 10

time took : 5.2089362144470215 with p value : 1 with iteration : 20

time took : 7.820969104766846 with p value : 1 with iteration : 30

time took : 10.427818775177002 with p value : 1 with iteration : 40

time took : 12.978275537490845 with p value : 1 with iteration : 50

time took : 15.522330045700073 with p value : 1 with iteration : 60

time took : 18.13934016227722 with p value : 1 with iteration : 70

time took : 20.79414439201355 with p value : 1 with iteration : 80

time took : 23.36881947517395 with p value : 1 with iteration : 90

time took : 25.955464124679565 with p value : 1 with iteration : 100
```

Figure 18. Empirical runtime of batched kernelized perceptron

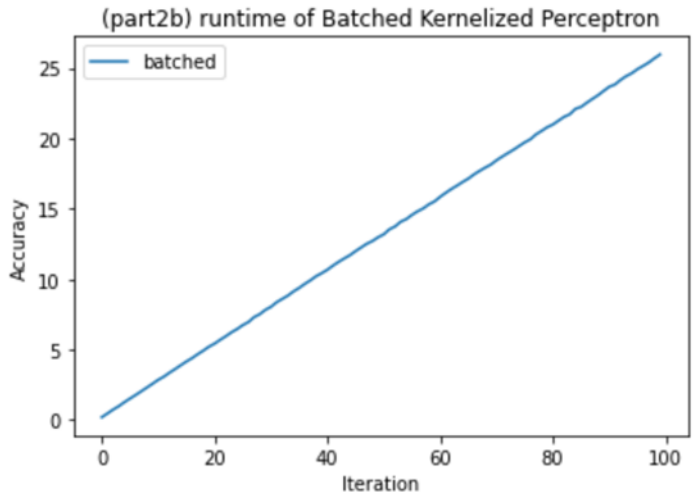

Figure 19. Empirical Runtime graph

In the figure 18 and 19, we can see the empirical runtime of batched kernelized perceptron algorithm. In figure 11, it shows the printed runtime result of 10^{th}, 20^{th}, … ,100^{th} iteration. As the iteration count gets bigger, we can see that the runtime of iteration gets longer. Also, in when we look at the figure 19, the runtime along the iteration gets longer, since the number of the iteration gets bigger.

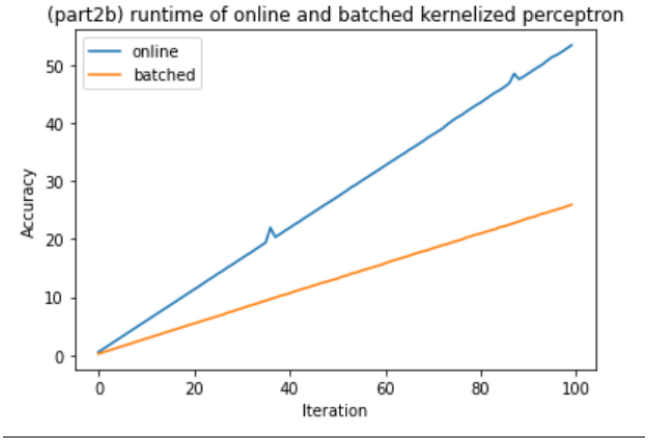

Figure 20. Online and Batched Kernelized perceptron runtime

In figure 20, we can see the empirical runtime plot of both online and batched kernelized perceptron. Both algorithms run longer as the iteration gets bigger but batched algorithm takes much less time compare to the online algorithm. This is because, online algorithm loop through each dataset and then make a prediction which results the complexity $O(n^3)$, and batched algorithm do the matrix calculation which results the complexity of $O(n^2)$. Therefore, batched kernelized perceptron takes less time to run the algorithm.