CS534 — Implementation Assignment 2

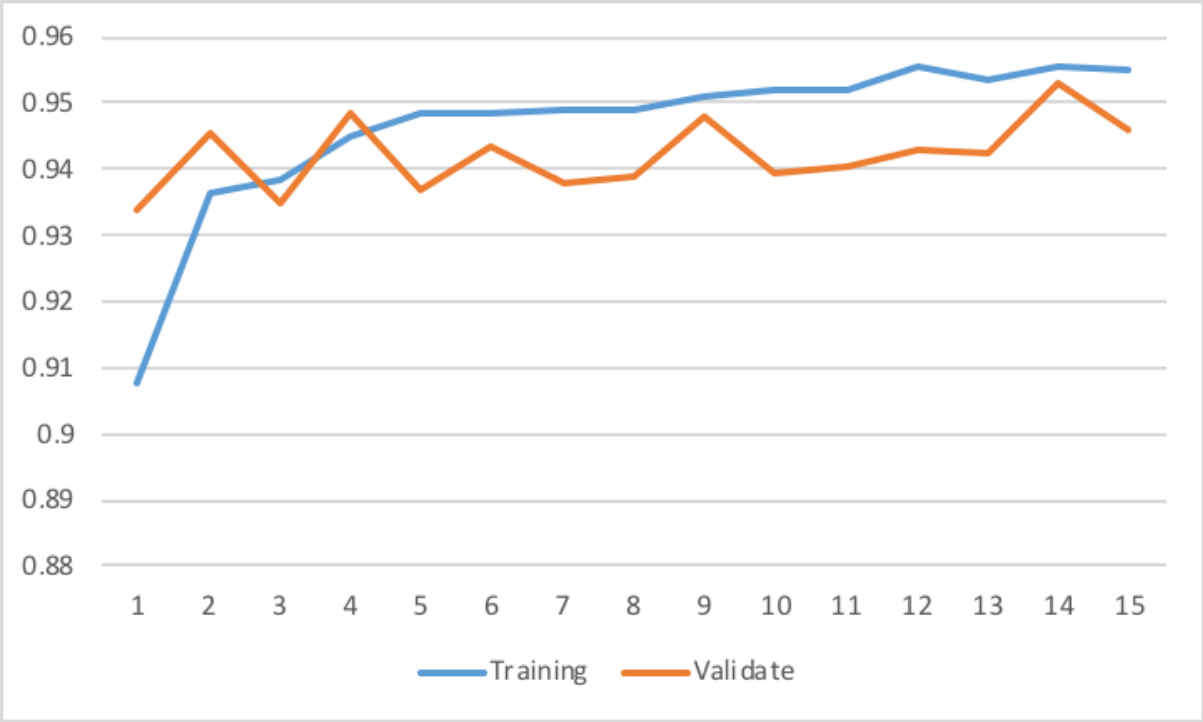
Chih-Hsiang Wang
Jui-Hung Lu
Webster Cheng

Contribution	Chih-Hsiang Wang	Jui-Hung Lu	Webster Cheng
Percentage (%)	33.3	33.3	33.4

General Introduction:

Part 1 (20 pts) : Online Perceptron.
(a) Implement the online perceptron model with algorithm described in Algorithm 1. Set the iters = 15. During the training, at the end of each iteration use the current w to make prediction on the validation samples. Record the accuracies for the train and validation at the end of each iteration. Plot the recorded train and validation accuracies versus the iteration number.

Iteration	Training	Validate
1	0.907733224	0.933701657
2	0.936170213	0.945365255
3	0.938216039	0.934929405
4	0.944762684	0.948434622
5	0.948240589	0.936771025
6	0.948445172	0.943523634
7	0.94905892	0.937998772
8	0.94905892	0.938612646
9	0.951104746	0.947820749
10	0.951718494	0.939226519
11	0.951923077	0.940454266
12	0.955605565	0.942909761
13	0.953559738	0.942295887
14	0.955605565	0.952731737
15	0.954787234	0.945979128



(b) Does the train accuracy reach to 100%? Why?

No, maybe it is because the dataset is not linear separable. Another possibility is that we cannot use online perceptron to find the model with largest margin, which constraints the growth of accuracy.

(c) Use the validation accuracy to decide the test number for iters. Apply the resulting model to make predictions for the samples in the test set. Generate the prediction file oplabel.csv. Please note that your file should only contain +1 (for 3) and -1 (for 5) and the number of rows should be the same as pa2 test.csv.

We decide to use weight values from 14^{th} iteration to predict for the samples in the test set. (The prediction is listed in oplabel.csv file)

Part 2 (20 pts) : Average Perceptron.

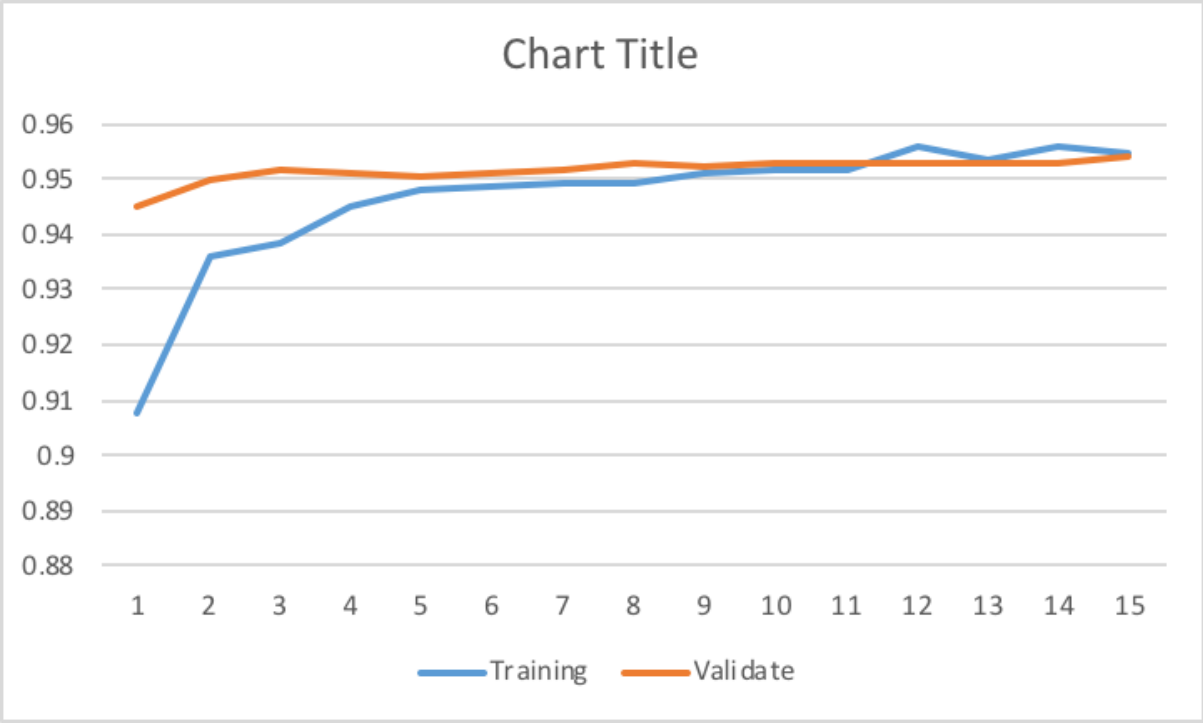
(a) Please implement the average perceptron described in Algorithm 2.

(implemented in the function avgPerceptron() in perceptron.py)

(b) Plot the train and validation accuracies versus the iteration number for iters = 1, ..., 15.

Iteration	Training	Validate
1	0.907733224	0.944751381
2	0.936170213	0.94966237
3	0.938216039	0.95150399
4	0.944762684	0.950890117
5	0.948240589	0.950276243
6	0.948445172	0.950890117
7	0.94905892	0.95150399

8	0.94905892	0.952731737
9	0.951104746	0.952117864
10	0.951718494	0.952731737
11	0.951923077	0.952731737
12	0.955605565	0.952731737
13	0.953559738	0.952731737
14	0.955605565	0.952731737
15	0.954787234	0.953959484



(c) How average model has affected the validation accuracy comparing to the online perceptron?
The validation accuracy of the average model is higher and smoother than the online perceptron.

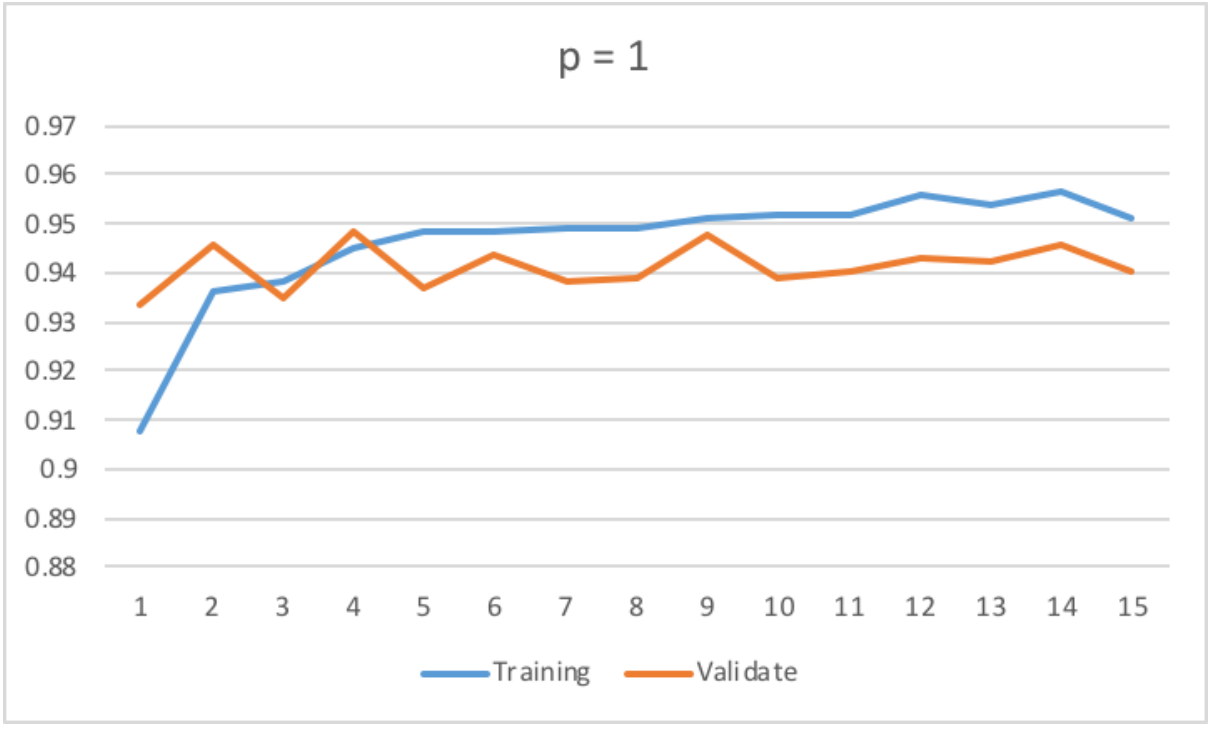
Part 3 (40 pts). Polynomial Kernel Perceptron.
(a) Implement the polynomial kernel function kp in the Algorithm 3. This function takes two vectors x1and x2 and an integer p for the polynomial degree, and returns a real value.
kp is implemented in the function ***setKMapValue*** in the ***kernelPerceptron.py*** file

(b) Define a Gram matrix K with size N × N where N is the number of training samples. Fill matrixK(i,j) = kp(xi,xj) for all of the pairs in the training set.
matrixK(i,j) = kp(xi,xj) is implemented in the function "***setKMapValue***" in the ***kernelPerceptron.py*** file

(c) Implement the rest of the kernel perceptron in Algorithm 3. For each p in [1, 2, 3, 7, 15]:
1) Run the algorithm to compute α.
2) At the end of each iteration use the current α to predict validation set.
3) Record the train and validation accuracy for each iteration and plot the train and validation accuracies versus the iteration number.

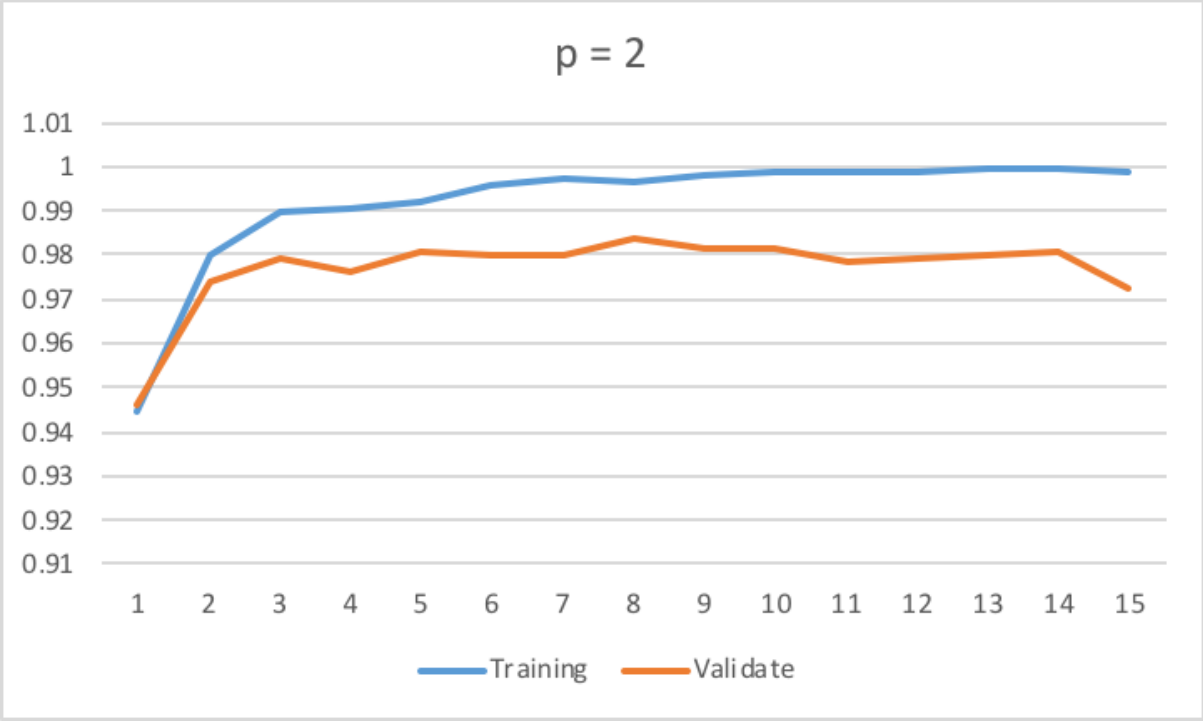
Polynomial degree = 1

Iteration	Training	Validate
1	0.907733224	0.933701657
2	0.936170213	0.945365255
3	0.938216039	0.934929405
4	0.944762684	0.948434622
5	0.948240589	0.936771025
6	0.948445172	0.943523634
7	0.94905892	0.937998772
8	0.94905892	0.938612646
9	0.951104746	0.947820749
10	0.951718494	0.939226519
11	0.951923077	0.940454266
12	0.955605565	0.942909761
13	0.953559738	0.942295887
14	0.956628478	0.945979128
15	0.951309329	0.940454266



Polynomial degree = 2

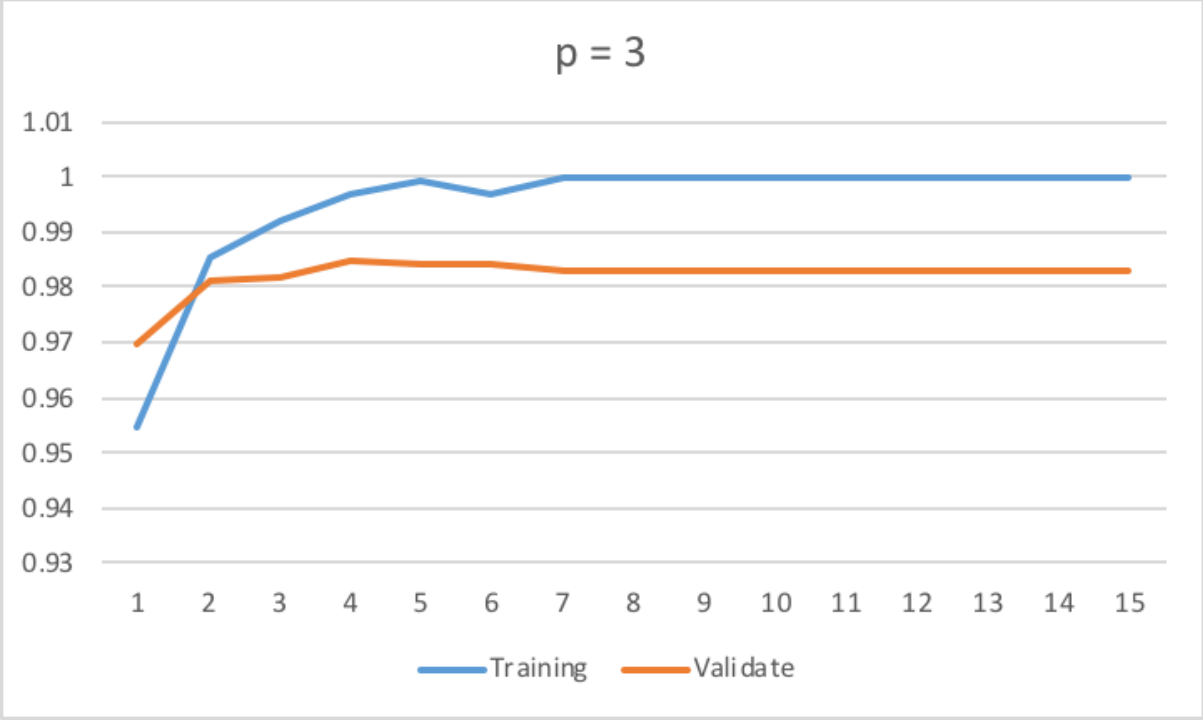
Iteration	Training	Validate
1	0.944762684	0.945979128
2	0.980155483	0.973603438
3	0.989770867	0.9791283
4	0.990384615	0.976058932
5	0.992021277	0.98096992
6	0.995703764	0.979742173
7	0.99693126	0.979742173
8	0.996317512	0.983425414
9	0.997954173	0.981583794
10	0.998977087	0.981583794
11	0.998567921	0.978514426
12	0.998977087	0.9791283
13	0.999386252	0.979742173
14	0.999181669	0.980356047
15	0.998567921	0.972375691



Polynomial degree = 3

Iteration	Training	Validate
1	0.95458265	0.969920196

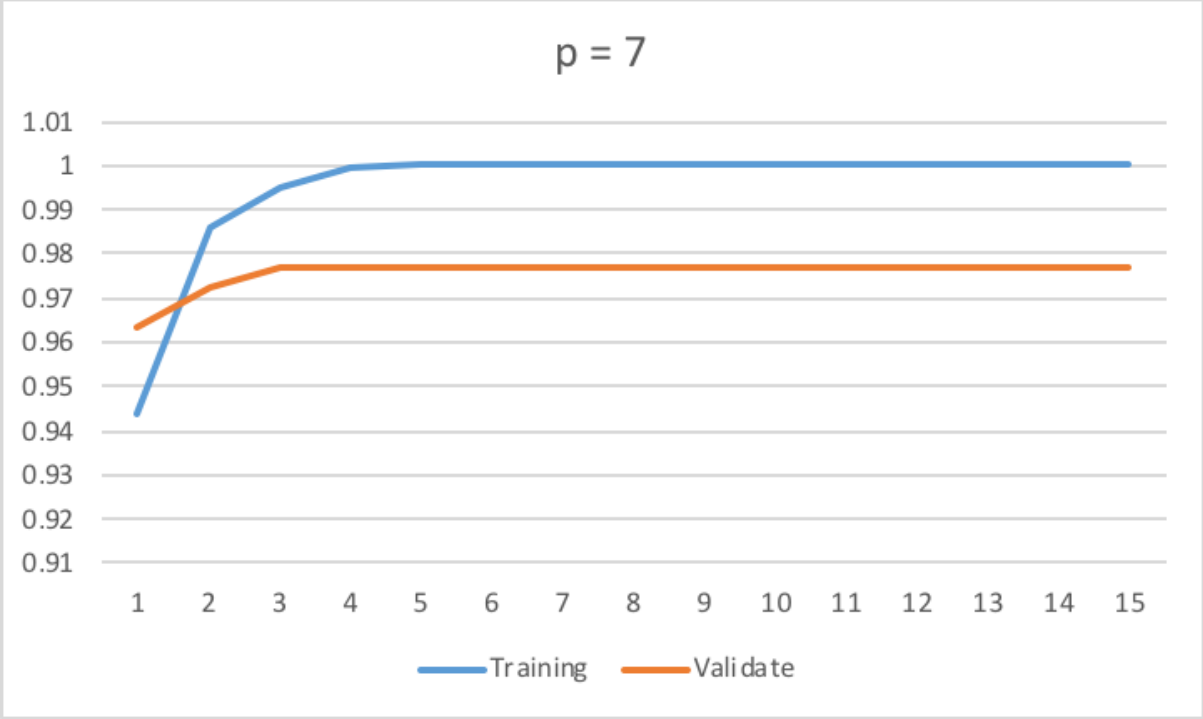
2	0.98527005	0.98096992
3	0.99181669	0.981583794
4	0.99652209	0.984653161
5	0.99918167	0.984039288
6	0.99652209	0.984039288
7	0.99979542	0.982811541
8	1	0.982811541
9	1	0.982811541
10	1	0.982811541
11	1	0.982811541
12	1	0.982811541
13	1	0.982811541
14	1	0.982811541
15	1	0.982811541



Polynomial degree = 7

Iteration	Training	Validate
1	0.94414894	0.963167587
2	0.9858838	0.972375691
3	0.9952946	0.977286679
4	0.99938625	0.977286679
5	1	0.977286679

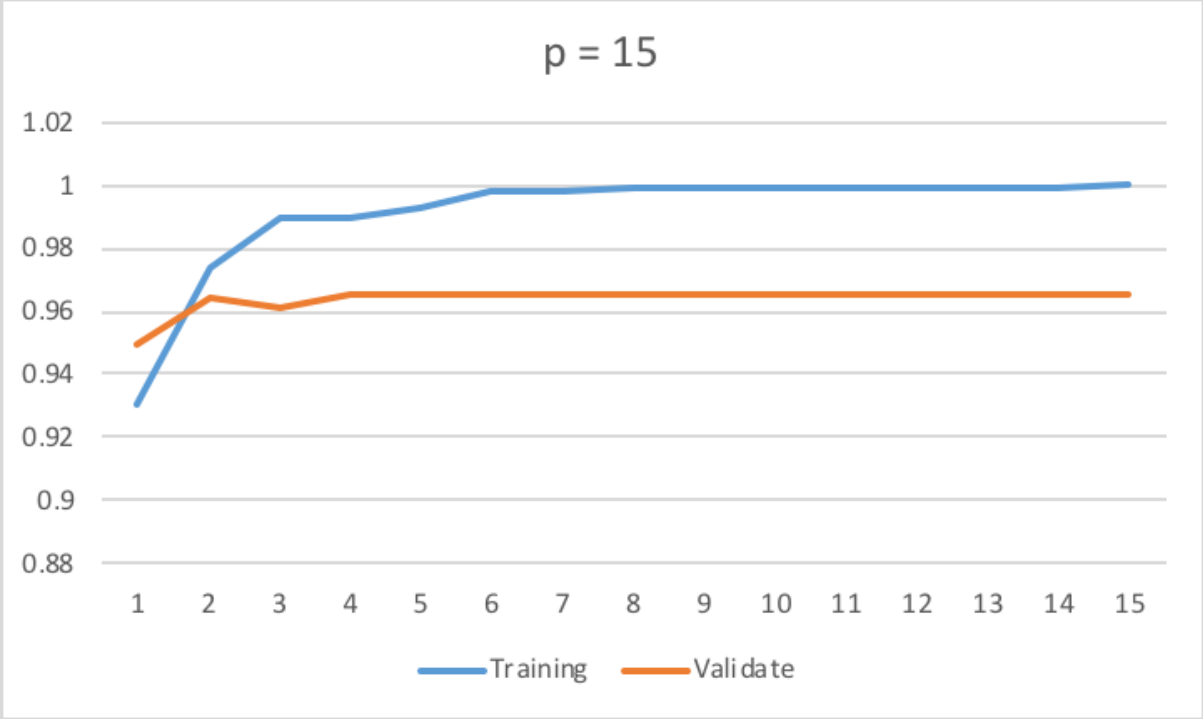
6	1	0.977286679
7	1	0.977286679
8	1	0.977286679
9	1	0.977286679
10	1	0.977286679
11	1	0.977286679
12	1	0.977286679
13	1	0.977286679
14	1	0.977286679
15	1	0.977286679



Polynomial degree = 15

Iteration	Training	Validate
1	0.930646481	0.94966237
2	0.973813421	0.964395335
3	0.98997545	0.961325967
4	0.989361702	0.965009208
5	0.992839607	0.965623082
6	0.997545008	0.965623082
7	0.997954173	0.965623082
8	0.998567921	0.965623082
9	0.998977087	0.965623082

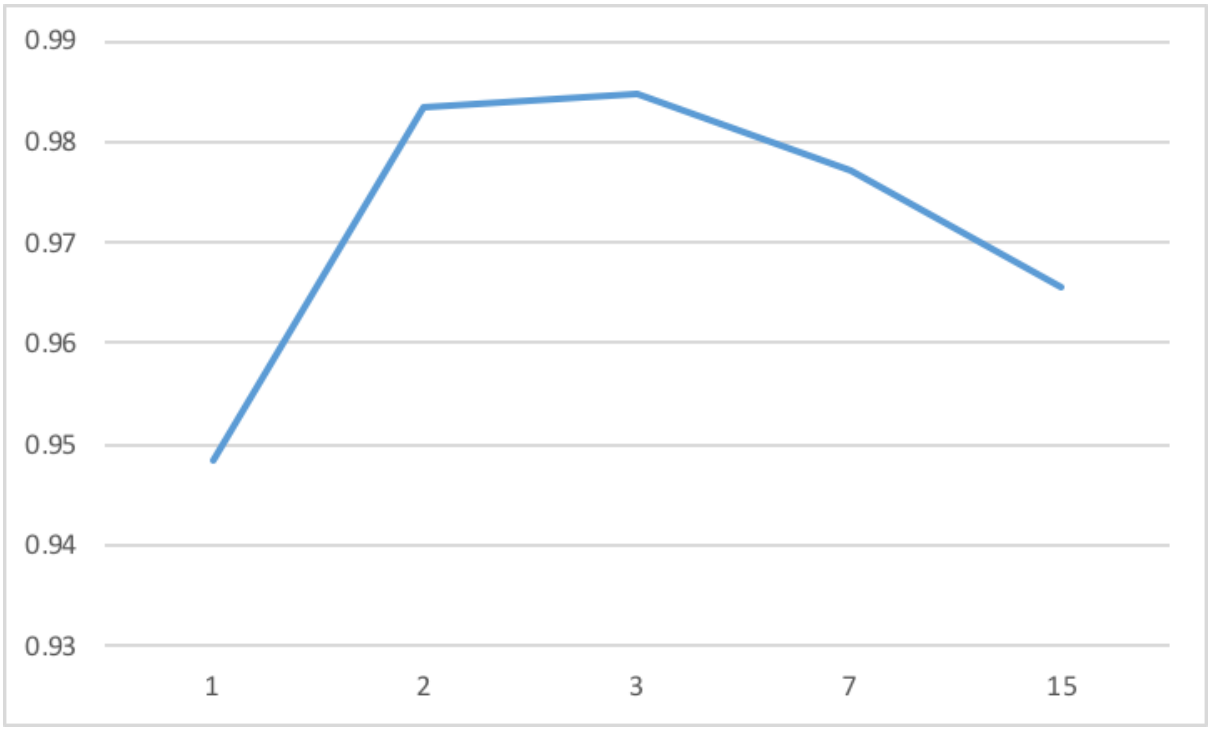
10	0.998977087	0.965623082
11	0.998977087	0.965623082
12	0.999181669	0.965623082
13	0.999181669	0.965623082
14	0.999386252	0.965623082
15	0.999590835	0.965623082



4) Record the best validation accuracy achieved for each p over all iterations.

p	best accuracy
1	0.948434622
2	0.983425414
3	0.984653161
7	0.977286679
15	0.965623082

(d) Plot the recorded best validation accuracies versus degrees. Please explain how p is affecting the train and validation performance.



The higher p create higher dimension space that finds the linear separable easily. Also, the value p is one of the ways to speed up the algorithm.
In the experiment, the higher p found the line to separate the data-set. This is different than p=1 and p=2 that cannot be linearly separable. Moreover, the value p is one of the ways to speed up the algorithm that finds the separable line quickly. This could be observed the result between p=3 and p=7 that the bigger value of p can become converged earlier.
However, when the algorithm speeding up too fast like p=15, it gots a little variation in the beginning instead. Therefore, the appropriate value of p can let our algorithm products the final accuracy faster.

(e) Use your best α (the best you found over all d and iterations above) to predict the test data-set. Please name the predicted file as kplabel.csv.

We choose p = 3 and iteration = 4 to predict the test data-set.