# (CS-534) Machine Learning
# Implementation Assignment #2

## Perceptron algorithm for Optical Character Recognition

**Submission By:**

**Akash Agarwal**
**OSU ID NUMBER:** 933-471-097
**Vishnupriya Nochikaduthekkedath Reghunathan**
**OSU ID NUMBER:** 933-620-571
**Anand P Koshy**
**OSU ID NUMBER:** 933-617-060

Contribution by each member in percentage : Equal contribution by each member

**Akash Agarwal - 33.33%**
**Vishnupriya Nochikaduthekkedath Reghunathan -33.33%**
**Anand P Koshy -33.33%**

### Introduction

In this assignment, we are using 3 variations of perceptron to to classify handwritten digits of numbers 3 or 5.

**Preprocessing**

1. The output Y from the train and validation datasets are changed to 1 and -1 ( +1 to 3 and -1 to 5) and removed from the dataset for further processing with the train and validation data.
2. Add bias feature ( equals to 1 ) to train, validation and test data to improve the fit.

**Part 1 : Online Perceptron**

In this part, we implemented the algorithm for Online Perceptron and trained the model with 15 iterations to calculate the weights per iteration. Then we used these weights on the validation data to calculate the validation loss. Recorded train and validation accuracies versus the iteration number is plotted to find the best iteration value. The weight and best iteration is used to calculate the output from test data and recorded in the file oplabel.csv.

**(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.**

**Solution**

| No of iterations | Accuracy with Training Data | Accuracy with Validation Data |
|---|---|---|
| 1 | 94.74222585924714 | 93.37016574585635 |
| 2 | 95.76513911620295 | 94.536525475752 |
| 3 | 94.98772504091653 | 93.49294045426642 |

| | | |
|---|---|---|
| 4 | 96.48117839607201 | 94.84346224677716 |
| 5 | 95.68330605564648 | 93.67710251688153 |
| 6 | 95.76513911620295 | 94.3523634131369 |
| 7 | 95.88788870703765 | 93.7998772252916 |
| 8 | 95.66284779050737 | 93.86126457949663 |
| 9 | 95.90834697217676 | 94.78207489257213 |
| 10 | 96.2152209492635 | 93.92265193370166 |
| 11 | 96.2152209492635 | 94.04542664211174 |
| 12 | 95.90834697217676 | 94.29097605893186 |
| 13 | 96.4607201309329 | 94.22958870472684 |
| 14 | 96.62438625204582 | 95.2731737262124 |
| 15 | 95.64238952536824 | 94.59791282995702 |

Algorithm is implemented and values are recorded for 15 iterations
The maximum accuracy observed on training data is **96.62438625204582** and
The maximum accuracy observed for validation data is **95.2731737262124**.

**(b) Does the train accuracy reach to 100%? Why?**
**Solution**
No . When we iterate over 15 iterations the maximum accuracy for train data that we observed is 96.62438625204582 at 14 th iteration.

Reason :
This is one of the weaknesses of online perceptron. Online perceptron counts later points more than earlier points. Suppose we have 100000 examples and the online perceptron learns a really good model for the first 90000 examples and does not require any update but it encounters an error on the 90001th example and the weights get updated, there is a possibility that this update may have an adverse effect on the first 90000 examples which were correctly classified and hence it is difficult to achieve a training accuracy of 100%.

It might be because the data may not be completely linearly separable . i.e) There might be some values that cannot be correctly classified as a 3 or 5. After the 14 th iteration the model is overfitting and hence the accuracy reaches 100 (For 1700 iterations and above). So early stopping (at 15 iterations) can be used to prevent overfitting.

(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 observed that the best value for iteration to be 14. At 14th iteration we observed the highest accuracy for validation data .i.e) 95.2731737262124 and after the 14th iteration overfitting occurs and the accuracy of the validation data decreases and accuracy of training data increases causing overfitting.
The output is recorded in oplabel.csv.

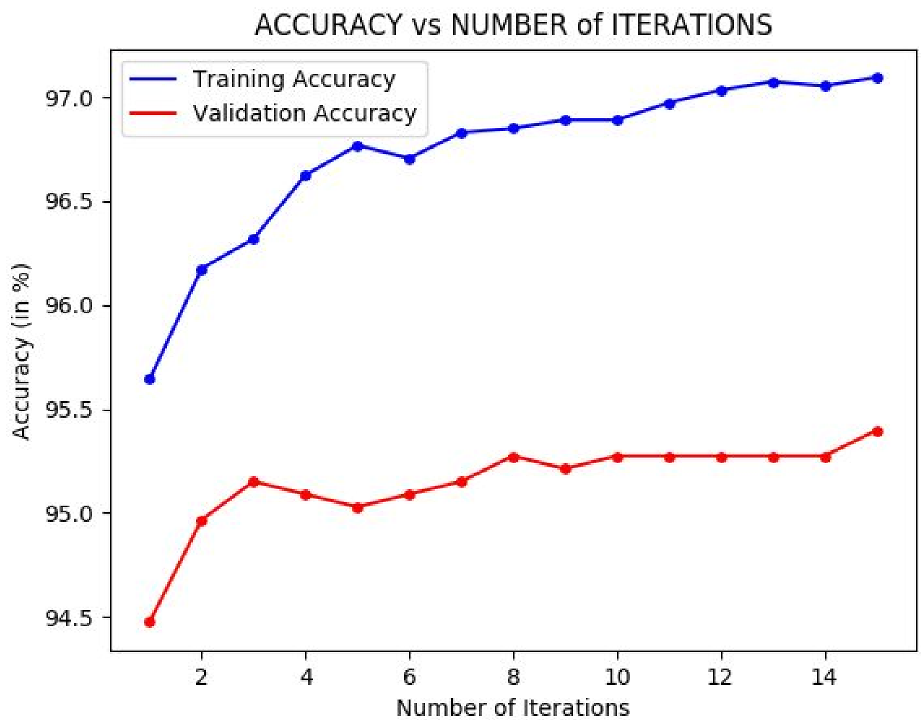
**Part 2 : Average Perceptron**

In this part, we implemented the average perceptron to improve the performance of the perceptron. Here we are taking the weighted average of all the intermittent weights to find the accuracy for new dataset (validation data) and observations are noted. The train and validation accuracies vs iteration are noted and then the validation accuracy is compared with the validation accuracy for online perceptron and observations are noted to show that average perceptron has better performance.
**(a) Please implement the average perceptron described in Algorithm 2**

Average Perceptron was implemented.

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

Plotted on the next page.




Algorithm is implemented and values are recorded for 15 iterations
The maximum accuracy observed on training data is **97.0949263502** and
The maximum accuracy observed for validation data is **95.3959484346**.

**(c) How average model has affected the validation accuracy comparing to the online perceptron?**

We observe that the validation plot for the average perceptron is smooth with less oscillations as compared to the validation plot for the online perceptron. (i.e) average perceptron gives more convergence ). The highest accuracy obtained on validation data using average perceptron (95.396) has a slight improvement over the highest accuracy reached by online perceptron (95.273) for 15th iteration. Even though the maximum accuracy obtained by average perceptron is same as online perceptron, the accuracy obtained by average perceptron at each iteration is observed to be almost similar for all the 15 iterations giving us a smooth curve.

| Iteration | Accuracy obtained by Online Perceptron | Accuracy obtained by Average Perceptron |
|---|---|---|
| 1 | 93.37016574585635 | 94.47513812154696 |

| | | |
|---|---|---|
| 2 | 94.536525475752 | 94.96623695518723 |
| 3 | 93.49294045426642 | 95.15039901780233 |
| 4 | 94.84346224677716 | 95.0890116635973 |
| 5 | 93.67710251688153 | 95.02762430939227 |
| 6 | 94.3523634131369 | 95.0890116635973 |
| 7 | 93.7998772252916 | 95.15039901780233 |
| 8 | 93.86126457949663 | 95.2731737262124 |
| 9 | 94.78207489257213 | 95.21178637200737 |
| 10 | 93.92265193370166 | 95.2731737262124 |
| 11 | 94.04542664211174 | 95.2731737262124 |
| 12 | 94.29097605893186 | 95.2731737262124 |
| 13 | 94.22958870472684 | 95.2731737262124 |
| 14 | 95.2731737262124 | 95.2731737262124 |
| 15 | 94.59791282995702 | 95.39594843462247 |

**Part 3 : Polynomial Kernel Perceptron**

**(a) Implement the polynomial kernel function $k_p$ in the Algorithm 3. This function takes two vectors $x_1$ and $x_2$ and an integer p for the polynomial degree, and returns a real value.**
Implemented

**(b) Define a Gram matrix K with size N × N where N is the number of training samples. Fill matrix K(i, j) = k p ($x_i$ , $x_j$ ) for all of the pairs in the training set.**
Implemented

**(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.**
We ran the algorithm and computed alpha for each iteration.

**2) At the end of each iteration use the current α to predict validation set.**
Completed

**3) Record the train and validation accuracy for each iteration and plot the train and validation accuracies versus the iteration number.**

**P = 1**

| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 94.74222586 | 93.37016575 |
| 2 | 95.76513912 | 94.53652548 |
| 3 | 94.98772504 | 93.49294045 |
| 4 | 96.4811784 | 94.84346225 |
| 5 | 95.68330606 | 93.67710252 |
| 6 | 95.76513912 | 94.35236341 |
| 7 | 95.88788871 | 93.79987723 |
| 8 | 95.66284779 | 93.86126458 |
| 9 | 95.90834697 | 94.78207489 |
| 10 | 96.21522095 | 93.92265193 |
| 11 | 96.21522095 | 94.04542664 |
| 12 | 95.90834697 | 94.29097606 |
| 13 | 96.46072013 | 94.2295887 |
| 14 | 96.56301146 | 94.59791283 |
| 15 | 95.45826514 | 94.04542664 |
| 16 | 96.64484452 | 94.04542664 |
| 17 | 95.88788871 | 93.86126458 |
| 18 | 96.80851064 | 94.9048496 |
| 19 | 96.29705401 | 94.35236341 |
| 20 | 95.17184943 | 93.30877839 |
| 21 | 94.82405892 | 93.24739104 |
| 22 | 95.92880524 | 93.30877839 |
| 23 | 96.23567921 | 95.02762431 |
| 24 | 96.4198036 | 94.35236341 |
| 25 | 96.68576105 | 94.59791283 |

**P = 2**

| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 96.44026187 | 94.59791283 |
| 2 | 98.2405892 | 97.36034377 |
| 3 | 99.26350245 | 97.91282996 |
| 4 | 98.91571195 | 97.60589319 |
| 5 | 99.67266776 | 98.09699202 |
| 6 | 99.67266776 | 97.97421731 |
| 7 | 99.83633388 | 97.97421731 |
| 8 | 99.95908347 | 98.34254144 |
| 9 | 99.77495908 | 98.15837937 |
| 10 | 99.95908347 | 98.15837937 |
| 11 | 99.87725041 | 97.8514426 |
| 12 | 99.87725041 | 97.91282996 |
| 13 | 99.95908347 | 97.97421731 |
| 14 | 99.83633388 | 98.03560467 |
| 15 | 99.32487725 | 97.23756906 |
| 16 | 99.85679214 | 98.09699202 |
| 17 | 99.83633388 | 98.03560467 |
| 18 | 99.97954173 | 98.15837937 |
| 19 | 99.85679214 | 98.21976673 |
| 20 | 99.95908347 | 97.97421731 |
| 21 | 100 | 97.97421731 |
| 22 | 100 | 97.97421731 |
| 23 | 100 | 97.97421731 |
| 24 | 100 | 97.97421731 |
| 25 | 100 | 97.97421731 |

**P = 3**

| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 98.07692308 | 96.99201964 |
| 2 | 99.28396072 | 98.09699202 |
| 3 | 99.83633388 | 98.15837937 |
| 4 | 99.89770867 | 98.46531614 |
| 5 | 99.97954173 | 98.40392879 |
| 6 | 99.95908347 | 98.40392879 |
| 7 | 100 | 98.28115408 |
| 8 | 100 | 98.28115408 |
| 9 | 100 | 98.28115408 |
| 10 | 100 | 98.28115408 |
| 11 | 100 | 98.28115408 |
| 12 | 100 | 98.28115408 |
| 13 | 100 | 98.28115408 |
| 14 | 100 | 98.28115408 |
| 15 | 100 | 98.28115408 |
| 16 | 100 | 98.28115408 |
| 17 | 100 | 98.28115408 |
| 18 | 100 | 98.28115408 |
| 19 | 100 | 98.28115408 |
| 20 | 100 | 98.28115408 |
| 21 | 100 | 98.28115408 |
| 22 | 100 | 98.28115408 |
| 23 | 100 | 98.28115408 |
| 24 | 100 | 98.28115408 |
| 25 | 100 | 98.28115408 |

**P = 7**

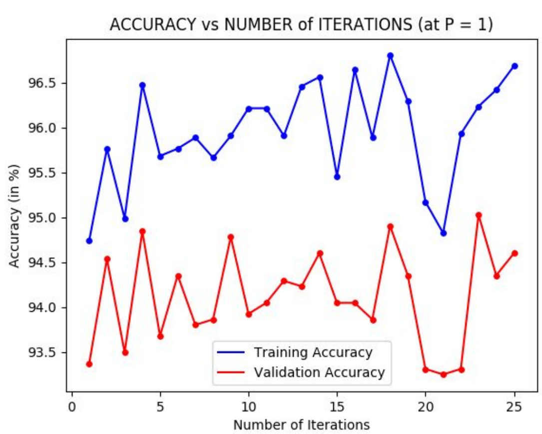
| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 98.79296236 | 96.31675875 |
| 2 | 99.61129296 | 97.23756906 |
| 3 | 99.9386252 | 97.72866789 |
| 4 | 100 | 97.72866789 |
| 5 | 100 | 97.72866789 |
| 6 | 100 | 97.72866789 |
| 7 | 100 | 97.72866789 |
| 8 | 100 | 97.72866789 |
| 9 | 100 | 97.72866789 |
| 10 | 100 | 97.72866789 |
| 11 | 100 | 97.72866789 |
| 12 | 100 | 97.72866789 |
| 13 | 100 | 97.72866789 |
| 14 | 100 | 97.72866789 |
| 15 | 100 | 97.72866789 |
| 16 | 100 | 97.72866789 |
| 17 | 100 | 97.72866789 |
| 18 | 100 | 97.72866789 |
| 19 | 100 | 97.72866789 |
| 20 | 100 | 97.72866789 |
| 21 | 100 | 97.72866789 |
| 22 | 100 | 97.72866789 |
| 23 | 100 | 97.72866789 |
| 24 | 100 | 97.72866789 |
| 25 | 100 | 97.72866789 |

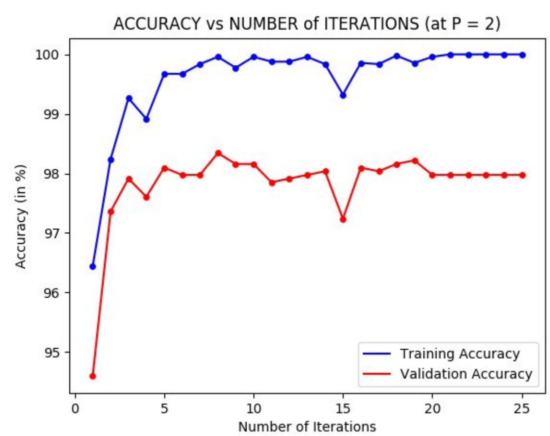
**P = 15**

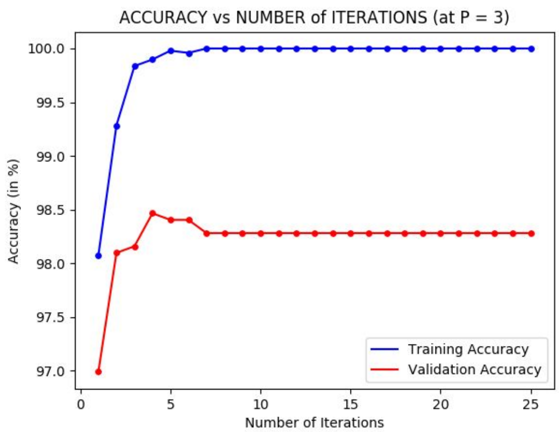
| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 96.70621931 | 94.96623696 |
| 2 | 99.24304419 | 96.43953346 |
| 3 | 98.93617021 | 96.13259669 |
| 4 | 99.03846154 | 96.50092081 |
| 5 | 99.75450082 | 96.56230816 |
| 6 | 99.79541735 | 96.56230816 |
| 7 | 99.85679214 | 96.56230816 |
| 8 | 99.89770867 | 96.56230816 |
| 9 | 99.89770867 | 96.56230816 |
| 10 | 99.89770867 | 96.56230816 |
| 11 | 99.91816694 | 96.56230816 |
| 12 | 99.91816694 | 96.56230816 |
| 13 | 99.9386252 | 96.56230816 |

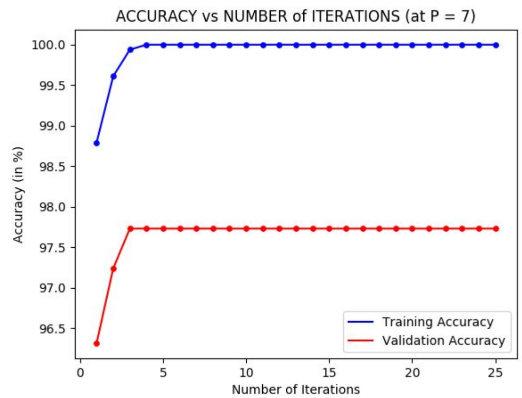
**P = 15**

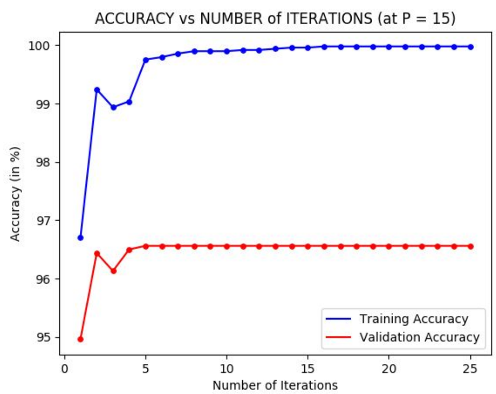
| Iteration | Training Accuracy | Validation Accuracy |
|---|---|---|
| 14 | 99.95908347 | 96.56230816 |
| 15 | 99.95908347 | 96.56230816 |
| 16 | 99.97954173 | 96.56230816 |
| 17 | 99.97954173 | 96.56230816 |
| 18 | 99.97954173 | 96.56230816 |
| 19 | 99.97954173 | 96.56230816 |
| 20 | 99.97954173 | 96.56230816 |
| 21 | 99.97954173 | 96.56230816 |
| 22 | 99.97954173 | 96.56230816 |
| 23 | 99.97954173 | 96.56230816 |
| 24 | 99.97954173 | 96.56230816 |
| 25 | 99.97954173 | 96.56230816 |





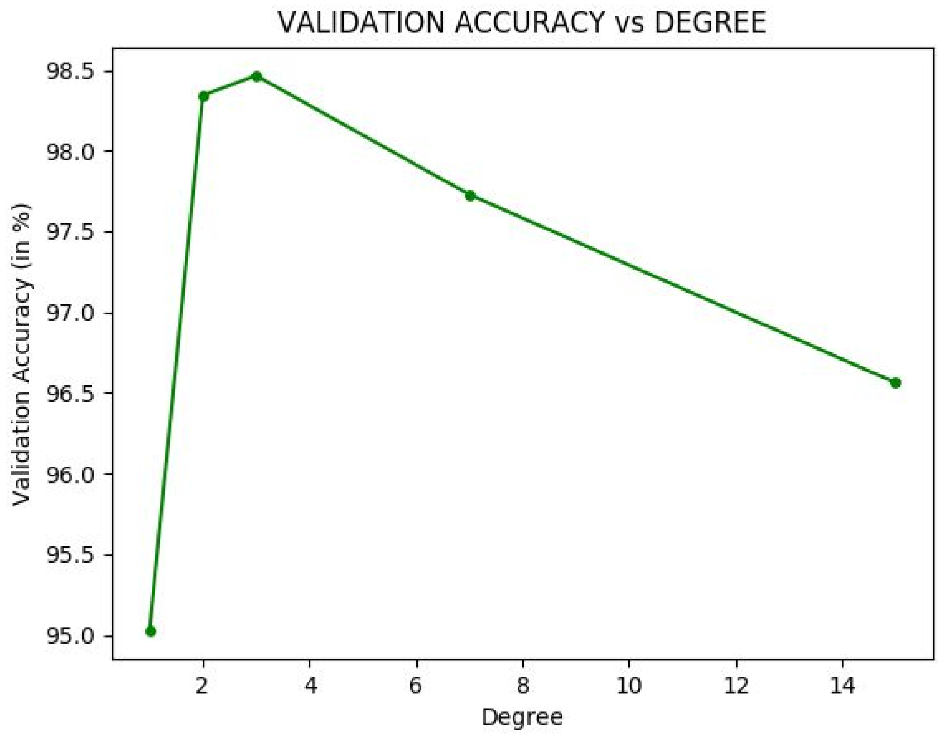




**4) Record the best validation accuracy achieved for each p over all iterations.**

| POWER | ACCURACY (in %) |
|---|---|
| P = 1 | 95.0276243094 |
| P = 2 | 98.3425414365 |
| P = 3 | 98.4653161449 |
| P = 7 | 97.7286678944 |
| P = 15 | 96.5623081645 |

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




It was observed that for Training examples, the accuracy increases as we increase the power. For example, for p = 1, the maximum training accuracy = 96.8 (approx), for higher values of p, the training accuracy is close to 100%. For p = 7, the model gains 100% accuracy from the 4th iteration. At p = 15, the training accuracy decreases very minutely, such that the maximum training accuracy at p equals 15 = 99.97% (approx).

For validation data, the values of validation accuracy increases as we increase the power, till p = 3. For values greater than p = 3, the validation accuracies starts to decrease.

Also, on increasing the degree of Kernelized- perceptron, it can be observed that the model gets fit to the data faster. For example, at p = 1, maximum validation accuracy is achieved at 24th iteration and for p = 3 it is achieved at 4th iteration. As we increase the power, we notice that overfitting occurs on the training data-set and hence the validation accuracy starts to decrease.

**(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.**

By observing the above graph, we can conclude that the best validation accuracy is observed at p = 3, i.e. cubic-kernelized perceptron will provide the best accuracy on the Validation Data for the given validation file. By Looking at the Alpha vs Iteration graph for p = 3, it was further observed that at Iteration: 4, the best value of alpha was observed. Thus, we've taken the alpha value for iteration:4 for power p = 3 to predict the Y values for the test data. The data has been recorded in kplabel.csv and has been submitted along with the report.