CS534 Implementation Assignment 2

November 2, 2019

Introduction

Optical Character Recognition is the problem of predicting a number (between 0 to 9) which best corresponds to a given image of a handwritten digit. In this assignment, we solve a simplified, binary version of this classification task in which we differentiate between images of threes and fives. We completed this task by implementing three variations of the perceptron algorithm: online, average, and kernel.

The data used in this assignment are a slightly modified version of Kaggle's digit recognizer dataset, filtered to include only images of threes and fives. We added additional preprocessing to make the data suitable for use with perceptrons. We changed the data labels from 3 and 5 to +1 and -1 respectively, and we introduced a bias term with a value of 1.0. Although shuffling data is good practice for perceptrons, we did not shuffle the data in order to produce consistent results.

This report is organized as follows. Part 1 compares the training and validation accuracy of an online perceptron after different numbers of training iterations. Part 2 compares the online perceptron to an average perceptron. Part 3 examines how train and validation accuracy for a polynomial kernel perceptron are affected by the use of different values of p.

Part 1: Online Perceptron

In this section, we train an online perceptron for 15 iterations and record the accuracy of the predictions generated by our model after every iteration.

(a) As shown in Figure 1, the train accuracy does not reach 100% after 15 iterations. There are two possible explanations for this result; either the perceptron was trained for an insufficient number of iterations or the data is not linearly separable. The latter explanation is more likely, given the lack of significant improvement in terms of train accuracy between iterations 4 and 15, as well as the sheer complexity of each of the points in the set.

(b) Figure 1 shows that the validation accuracy is highest after 14 iterations. We used the weights found after this number of training iterations to make predictions for pa2_test.csv, which can be found in oplabel.csv.

Part 2: Average Perceptron

In this section, we train an average perceptron for 15 iterations and record the accuracy of the predictions generated by our model after every iteration.

a) Figure 2 shows the train and validation accuracy after $1, 2, \ldots, 15$ iterations.

b) Compared to the online perceptron, the average perceptron shows a smooth increase in both train and validation accuracy as the number of iterations increases. These results are not surprising given that, because the average perceptron updates averaged weights, it reduces the tendency of the online perceptron to "overcorrect" misclassified examples.

We also note that both the training and validation accuracies end up higher in magnitude with the average perceptron compared to the online perceptron. This is interesting given that there's no guarantee that the averaging would result in higher accuracy; this seems to suggest that the tendency of the online perceptron to overcorrect is quite significant and detrimental to accuracy.

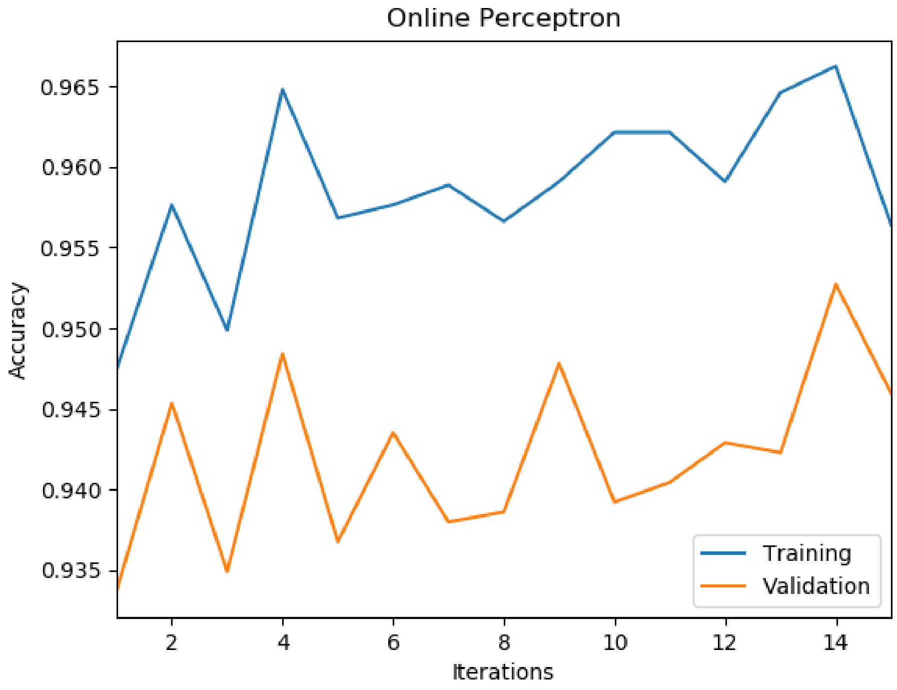


Figure 1: Problem 1a. Accuracy vs. Iteration of Online Perceptron

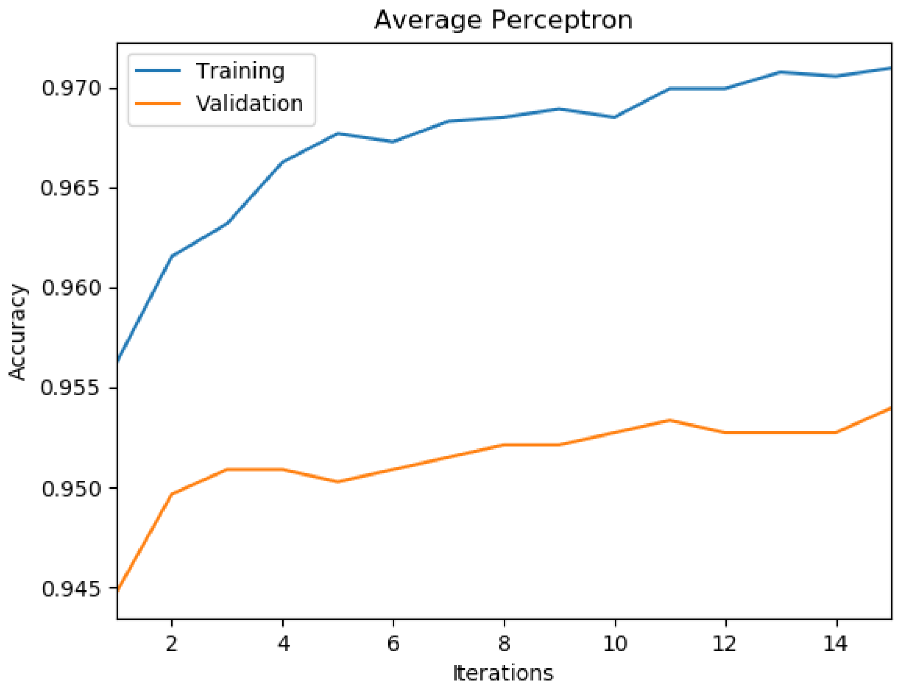
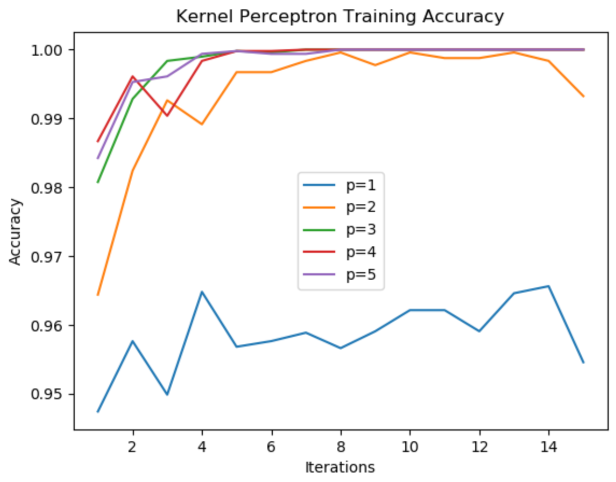


Figure 2: Problem 2a. Accuracy vs. Iteration of Average Perceptron

c) Figure 2 shows that the value for *iter* with the highest validation accuracy is 15. We used the weights found after this number of training iterations to make predictions for pa2_test.csv, which can be found in aplabel.csv.

Part 3: Polynomial Kernel Perceptron

In this section, we trained kernel perceptrons for 15 iterations and recorded the accuracy of the predictions generated by our model after every iteration. We used polynomial kernels, $k_p(x_1, x_2) = (1 + x_1^T x_2)^p$.



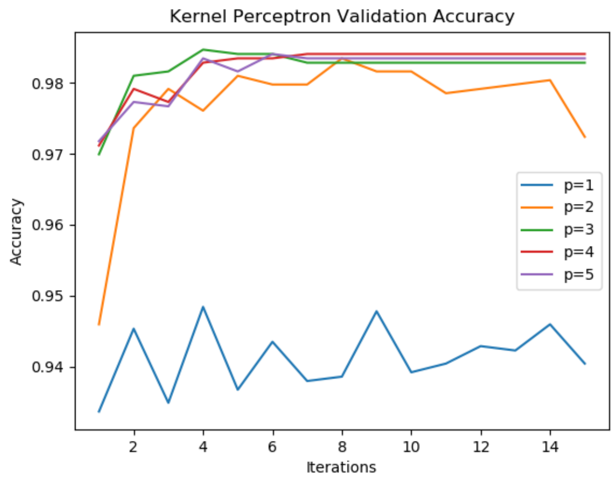


Figure 3: Problem 3a & 3b. Accuracy, Iteration, and p of Kernel Perceptron

a) Figure 3 shows the train and validation accuracy after $1, 2, \ldots, 15$ iterations for polynomial kernels with $p = 1, 2, \ldots, 5$. Figure 4 shows the best validation accuracy per p. As shown in these Figures, the best train accuracy increases with p until it reaches 100% at $p = 3$; $p = 4$ and 5 also attain 100% training accuracy. This means that our training data is, in fact, linearly separable in a higher dimensional space. It also makes sense that all models with $p \geq 3$ achieved perfect training accuracy since as we increase p, the corresponding feature space mapping Φ becomes higher dimensional; a data set which is separable in a lower-dimensional feature space (such as $p = 3$) is also likely to be separable in a higher-dimensional feature space (such as $p = 5$).

   Regarding the validation accuracy, we note that we actually attained the highest validation accuracy with $p = 3$, with a 98.47% accuracy. The fact that $p = 4$ and 5 did not perform better is because of overfitting; their feature space mappings Φ may contain many extraneous features that do not help classify real examples and simply result in overfitting.

b) In this experiment, the best kernel perceptron model was $p = 3$ after 4 iterations, and we made the test set prediction with it. The results are shown in kplabel.csv.

Conclusion

We found that the Average Perceptron shows a smooth increase the accuracy rate per iteration, while the Online Perceptron shows fluctuations in accuracy. Both algorithms never reach 100% accuracy, but the average perceptron shows a higher maximum accuracy rate. The polynomial Kernel Perceptron does reach 100% accuracy for the training set when $p \geq 3$. We found that the value of p affects the train and validation performance, and p=3 after four iterations is our best kernel perceptron model.

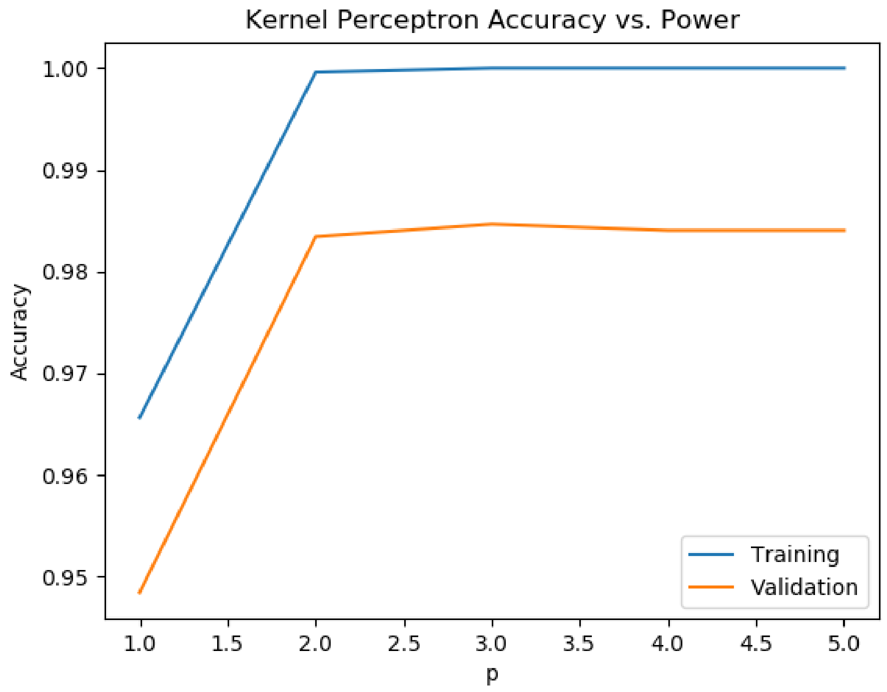


Figure 4: Problem 3a(2). Highest accuracy attained for training/validation for each p model.