Introduction

The goal of this assignment is to correctly classify a given image of a handwritten digit. The two classes are “3” and “5”. In our analysis these were the classes “+1” and “-1” respectively. The data comes from the MNIST Database via kaggle. Each sample is a list of 784 floating values which come from the flattened gray-scale values from a 2-d digital handwritten image. The train and validation set contain the target, and have 4888 and 1629 samples, respectively. The test set has 1629 samples. In this report we document our exploration of: three variations of the perceptron algorithm: online, average and polynomial kernel.

Part 1: Online Perceptron

(a)

Using the online perceptron we did not see the train accuracy reach 100%. The accuracy for both train and validation were highest after the fourteenth iteration. The training data had higher accuracy rates than the validation data set. Figure 1 shows the accuracy for the online perceptron.

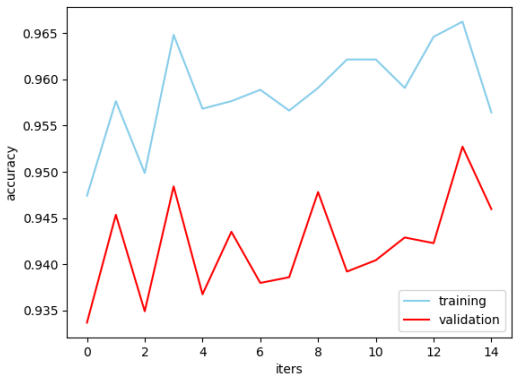

Figure 1: Accuracy for Online Perceptron

(b)

Using the validation accuracy, we find that the best value for *iters* is at 14. This was the case with the train and test sets.

Part 2: Average Perceptron

(a)

We implement the average perceptron on the train data. Figure 2 show the accuracy for the average perceptron on train and validation sets.

(b)

It seems like the accuracy curves for average are more stable than for the online perceptron, closer to being monotonic. This could be because as we expect the average perceptron to start doing better, the weights will not update for each sample. Eventually we would expect to update the average weight vector when the weight vector has changed substantially.

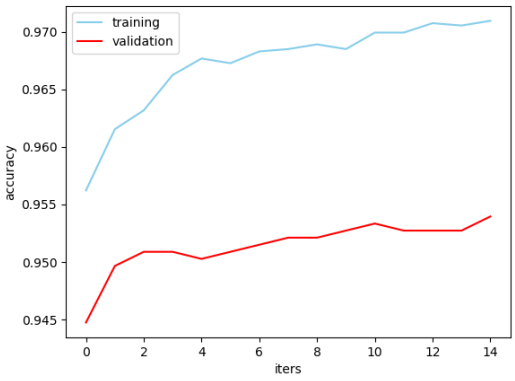

Figure 2: Accuracy for Average Perceptron

(c)

Using the validation accuracy, we find that the best value for *iters* is at 15. This was the case with the train and test sets.

Part 3: Polynomial Kernel Perceptron

(a)

We implement the kernelized perceptron for different degrees of polynomials. We record the train and validation accuracy and present them in Figure 3. We notice that the accuracy improves as the degree of the polynomial increases. Though this can lead to over fitting. This is displayed in Figure 4.
We record and plot the best accuracy for the validation data as a function of p. We notice that the accuracy for the training is higher than for validation. It approaches closest to perfect classification as the degree p increases. We conjecture that higher order polynomial kernels will perfectly separate the training data. With the validation data, increasing the degree p improves the accuracy, for a certain amount of time. This is displayed in Figure 4 as the polynomial with degree is 4 reaches the highest accuracy.

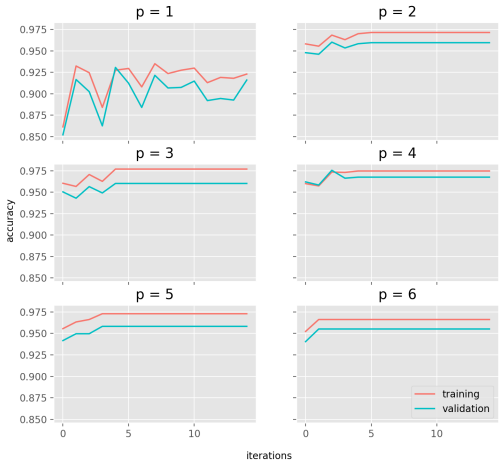

Figure 3: Kernel Perceptron Accuracy for different polynomial degrees

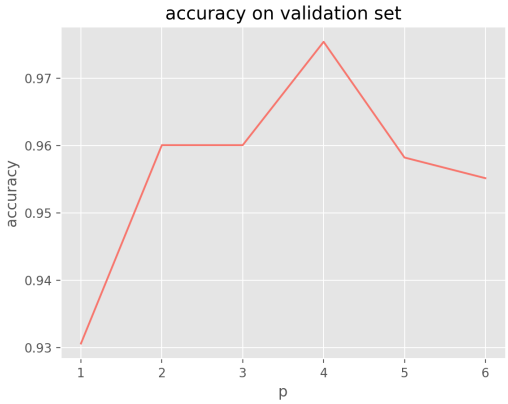

Figure 4: Kernel Perceptron Accuracy on different values of p

(b)

Using the best model, we make predict target values for the test set. These and other predictions are included in submission.

Discussion

We observed that even though it is possible to use high degrees of polynomials for the Polynomial Kernel Perceptron, it may not be the best strategy for generalization. The higher the degree of the polynomial, we can perfectly separate the training data. This was not the case with the validation data. Between the results of the online and average perceptrons, it seems that the online perceptron varies more in accuracy rates than the average perceptron. The average perceptron performs better than the online perceptron and the the polynomial kernel perceptron performs better than the other two.