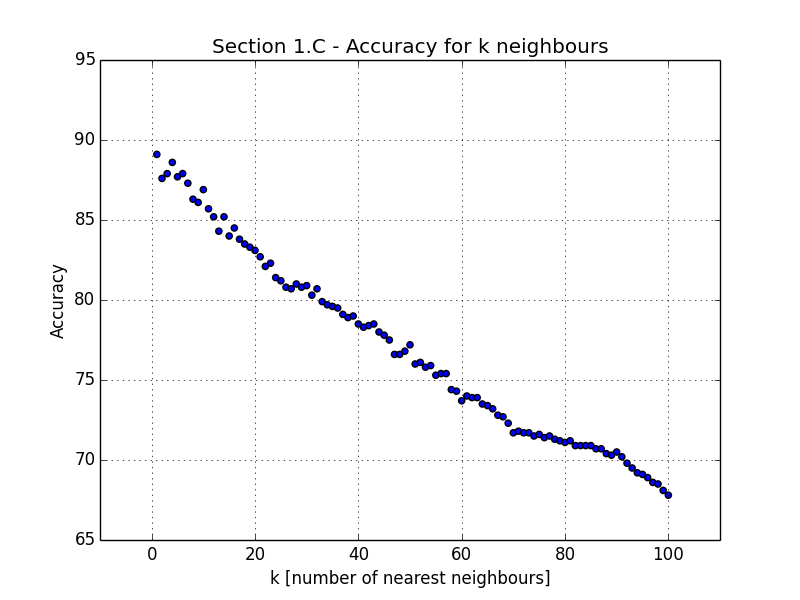
Assignment #3 – Programming Part

**Code Location:**

**Question 1**

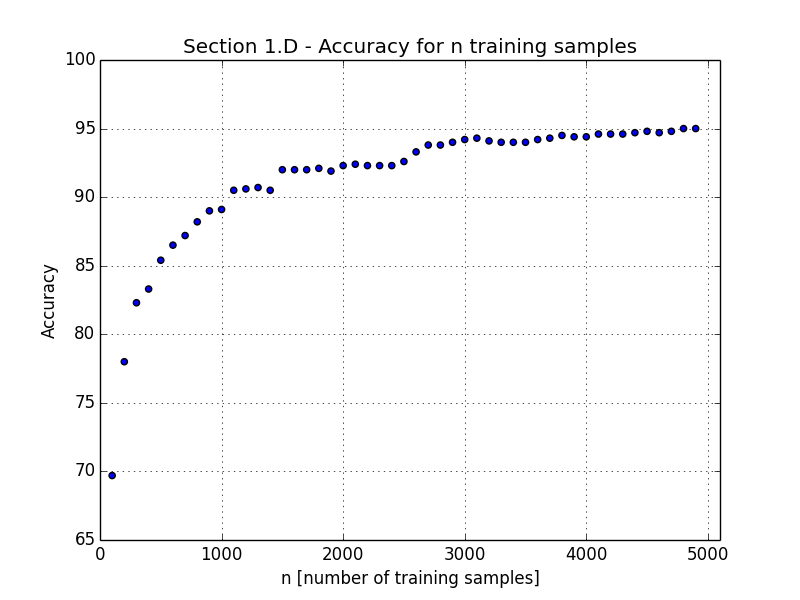
b. The accuracy we got is 86.9% on the test set.

We would expect from a random predictor to have a 10% accuracy, because the probability of correct prediction on one image is so in expectation it will get the correct label on 0.1 of the images.

c. As the plot suggests, as k increases the accuracy asymptotically decreases. Thus, knn with k=1 has the best accuracy.

The low k value may result from the very structure of the dataset. Most digits are pre-processed and centered, and therefore it’s acceptable to assume that for most digits we may find at least a single “twin” digit in the a training set that appears similar in nature.

A higher k implies lower accuracy. This result may occur because when taking more neighbors there is a bigger chance we will get more outliers as nearest neighbors. This may be because the l2 norm is imprecise for pixel-wise comparison (for example, for two identical digits with one of them shifted 2 pixels to the left, the l2 distance may be high), so the more samples taken into account – the more “outliers” may affect the final classification.

d. 

As n increases, the support in the feature space becomes more dense. Therefore, the chance of finding a “twin” for a test sample is higher and we get a more accurate prediction for the test samples in total.

The graph doesn’t grow linearly because at some point we have enough samples that give a good approximation for the distribution of the data in the feature space.

**Question 2**

1. The result:

n Mean Accuracy 5% 95%

---- --------------- ------- -------

5 88.94 83.6745 92.784

10 91.36 88.2293 94.6264

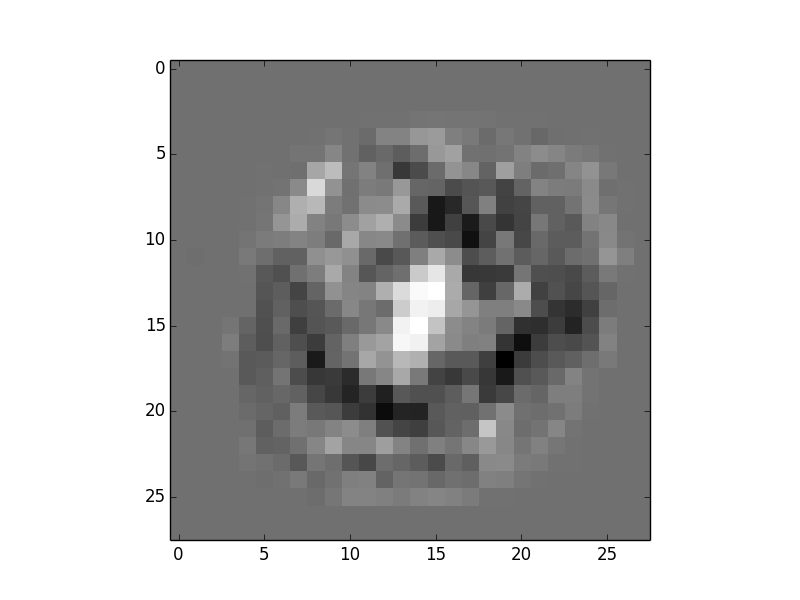
50 95.82 92.8147 97.8045

100 96.87 95.3327 98.1576

500 98.01 95.7446 98.9765

1000 98.62 97.935 99.0814

1. 98.84 97.7968 99.3347
2. The result:



The perceptron gives a high weight to the center of the image, where the ‘8’ figures have high response compared to the ‘0’ figures. Moreover, it gives lower weights to the circular shape in the middle, where the ‘0’ figures have higher response compared to the ‘8’ figures. It seems that most ‘0’ figures lean towards the right hand side.

The frame of the matrix gets neutral weights (gray) because it doesn’t help the perceptron to predict either way.

1. After training on the entire training dataset we got an accuracy of 98.52% over the test set.

Compared to the results in section A, considering a small error margin due to a single iteration, we get high accuracy over the test set since the perceptron have learned from enough examples.

1. Observe the following misclassifications:

|  |  |
| --- | --- |
| Macintosh HD:Users:orperel:Desktop:deep_tests:ml:IntroToML:Ex3:Section 2.D - Wrong prediction #80_label_-1.png | Macintosh HD:Users:orperel:Desktop:deep_tests:ml:IntroToML:Ex3:Section 2.D - Wrong prediction #506_label_1.png |
| *Sample #80 in test set* | *Sample #506 in test set* |

The perceptron fails to classify correctly both samples. Looking at the weight matrix obtained above, we expect pure white areas of the weight matrix to respond strongly for “1” labeled samples (e.g: “8” digits) and pure dark areas to respond strongly to “-1” samples (e.g: “0” digits). Comparing to the misclassified samples, it’s safe to assume that due to the “0” digit being too narrow, it falls short of the “black wide circular area”, and therefore the perceptron’s weights in the digit’s area aren’t salient enough for classification. A similar explanation can be attributed to the “8” digit which is quite wide compared to other “8” digits, and therefore responds to the weights that normally classify a zero digit (the middle edge that connects 8 digits is also “misplaced” related to most 8 digits, which is why this digit doesn’t activate the middle “white” area of the Perceptron’s weight matrix very strongly).

These examples can testify on the vulnerability of a pixel-wise loss and the limitations of the Perceptron, being to sensitive to samples that lie very close to the hyper-plane of separation between the 2 classes (the misplaced pixels aren’t robust enough as features for the Perceptron to classify deformed digits).