CS5785 Homework 1

Irene Font Peradejordi - if76, Antonio Mojena - am3238

September 26, 2019

1 Digit Recognizer

1.1 Plotting Data

Once the training data was downloaded from Kaggle, it was found that each row contained the label as its first element, and an 8-bit value for the remaining 784 elements. The latter was taken, reshaped to a numpy array of dimensions 28x28 and plotted.

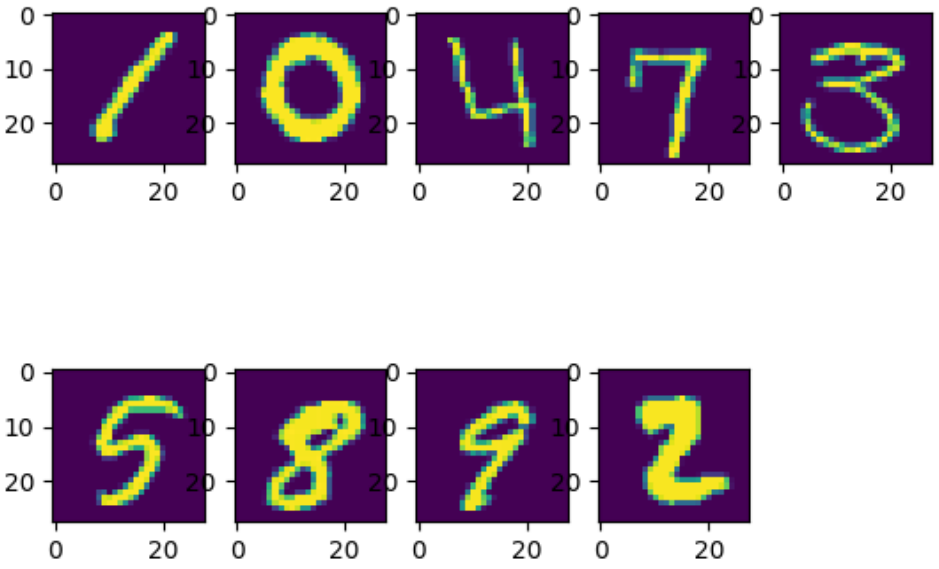

Figure 1: Each MNIST digit

1.2 Prior Probability

The prior probability for each label was calculated and plotted in a normalized histogram. As seen below, the distribution is uniform because each class

has roughly the same probability.

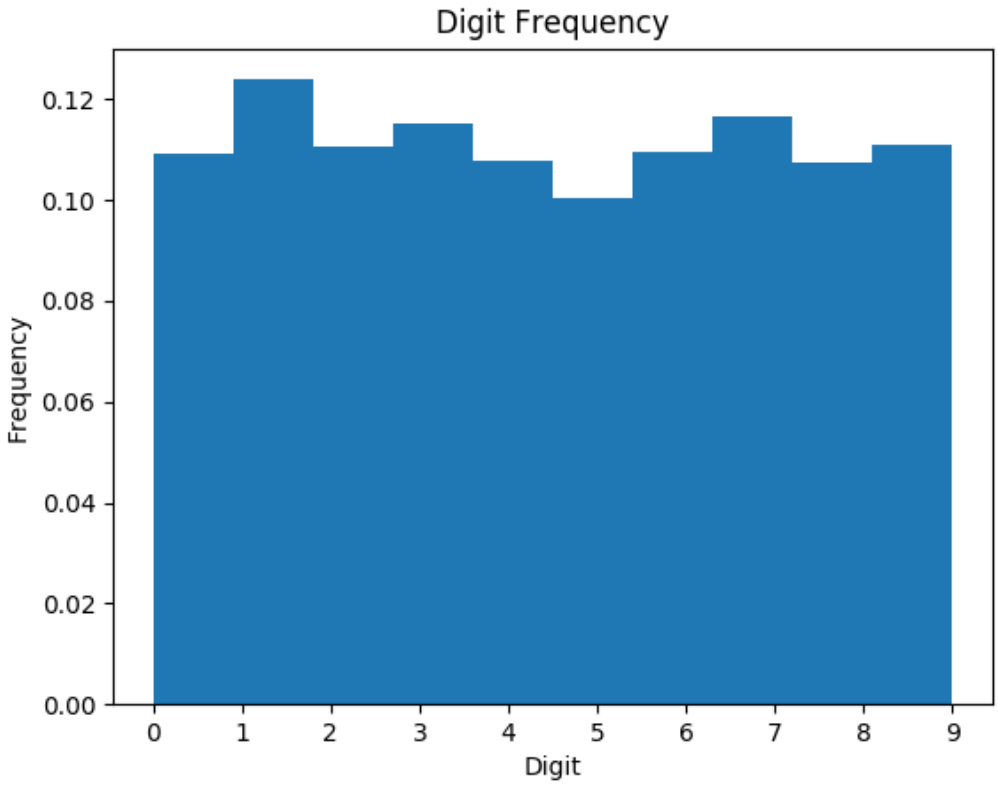

Figure 2: Digit Frequency in Testing Data

1.3 Nearest neighbor

The next task was to find one of each digit's best match. The best match as of this point is defined as its "closest neighbor", the element that has the closest values altogether for the digit in question. The results are shown below.

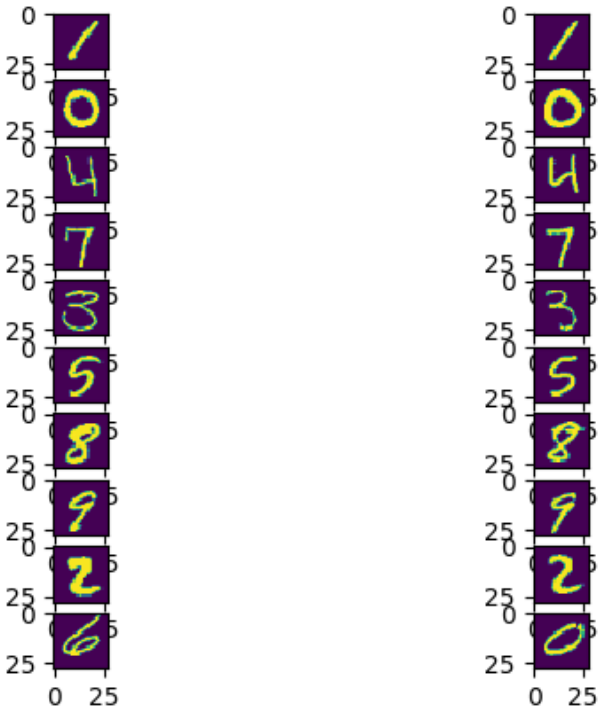

Figure 3: Best Match for a Single Digit

The digits on the left represent the digit whose best match we were trying to find; the digit is the best match. As you can see, this strategy works for the most part. There was only one erroneous match: a *0* was found to be the best match for a *6*.

1.4 Binary Comparison

The following task was to calculate all distances for all 0's and 1's in the training data set. This was done and the distances were classified as *genuine* or *impostor* distances. Genuine distances are all the distances between 0's with themselves and 1's with themselves. Any distance between 0's and 1's and vice versa is classified as an impostor distance. After the classification,

the distances were plotted in a normalized histogram to have a visual representation of the magnitude of each classification (i.e the largest genuine distance or shortest impostor distance). The figure below is an example of this visual representation on a subset of data.

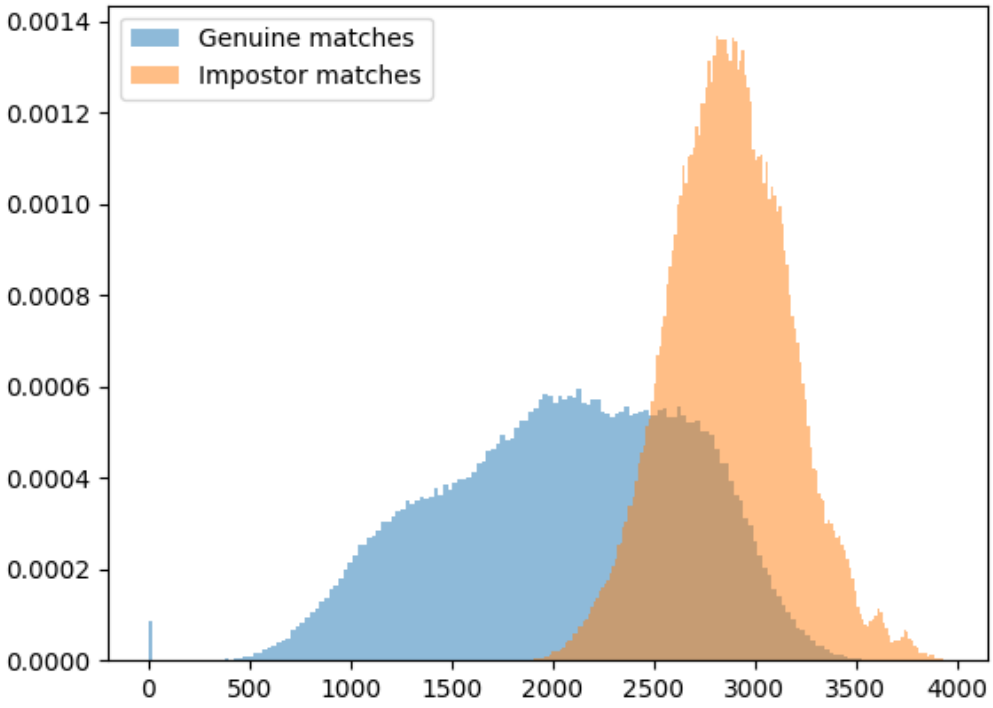

Figure 4: Binary Comparison and Classification of Distances

In this data subset, you can see the genuine distances have are in the range ˜ [500, 3500) and the impostor distances are in the range ˜ [2000, 4000). There is also a clear overlap between these classifications which explains why in general, this strategy could cause a mismatch between two digits.

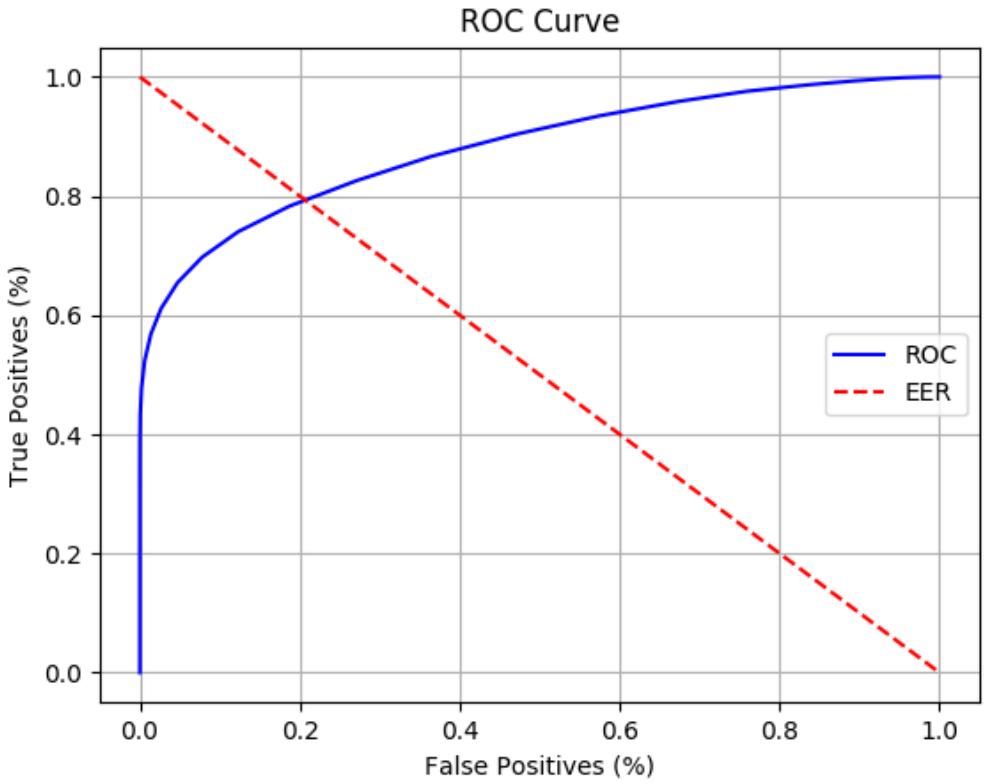

Figure 5: ROC

1.5 ROC Curve

After classifying the distances as either genuine or impostor distances. The ROC curve was calculated to find the equal error rate (EER) which, by inspection, is ˜ 0.2.

1.6 KNN Implementation

The KNN implementation was written calculating Euclidean distances as described in the sections above and comparing the k-nearest distances and classifying the test input by the mode of the classes predicted for the digit in question. Our output was submitted to Kaggle and received a 96.9% accuracy.

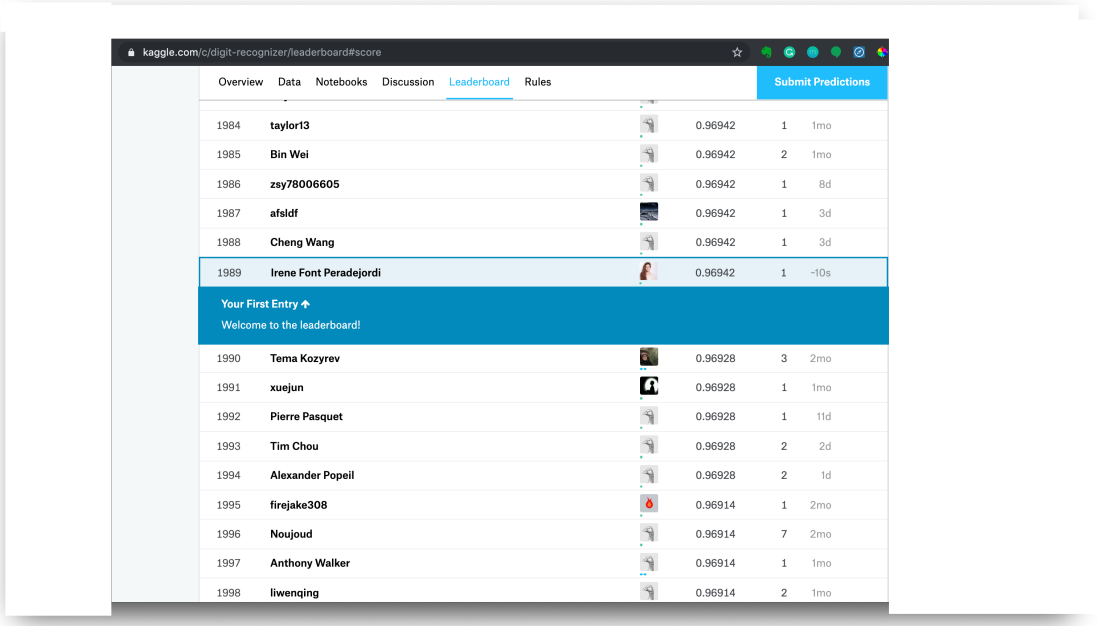

Figure 6: Kaggle Accuracy Report

1.7 Cross Validation

The cross validation was used to test the KNN implementation with 3 folds. The average accuracy was 95% with the following confusion matrix:

	0	1	2	3	4	5	6	7	8	9
0	1436	1	0	0	0	2	8	1	1	0
1	0	1689	2	1	0	1	4	3	1	1
2	19	28	1403	12	4	1	4	30	6	7
3	7	10	14	1430	1	33	4	11	12	5
4	2	18	2	0	1398	0	4	3	0	49
5	11	8	3	34	6	1269	12	1	5	9
6	5	2	0	0	1	8	1472	0	2	0
7	2	33	8	4	10	0	0	1466	0	27
8	11	26	14	37	7	32	10	5	1284	22
9	7	5	5	12	32	3	2	32	1	1387

Figure 7: Confusion Matrix

The green cells represent the number of matches right by digit. The red cells represent the mismatches that had a noticeable frequency. The common mismatches are:

- 2's –> 1's and 7's
- 3's –> 5's
- 4's –> 9's
- 5's –> 3
- 7's –> 1's and 9's
- 8's –> 3's and 5's
- 9's –> 4's and 7's

2 The Titanic Disaster

From all the features provided in the Titanic data set only some were used to train our logistic regression. These were chosen depending on their correlation with the passengers survival rate.

For instance, *Name*, and *Ticket* where immediately discarded for their insignificance. Neither the passengers name nor their ticket are relevant when predicting whether a passenger survived the disaster. *Cabin* was also discarded due to the amount of NaN values.

To understand the the strength of the relationship between the other features and our dependent variable (survival rate), the correlation coefficients were computed. These are the results:

- Survived vs Pclass: -0.33
- Survived vs Sex: -0.54
- Survived vs Age: -0.06
- Survived vs SibSp: -0.03
- Survived vs Fare: 0.25
- Survived vs Parch: 0.08
- Survived vs Embarked: 0.003

As seen above, *Sex* and *Pclass* have the stronger correlation. All the other features were used for the training except the *Embarked*, as we concluded his correlation is almost insignificant.

3 Written Exercises

3.1 Variance of a sum

Prove var [X − Y] = var[X] + var[Y] − 2cov[X,Y]

$$var[X-Y] = E[(X-Y)(X-Y)] - E[X+Y]^2$$

$$var[X-Y] = E[x^2 - 2XY + Y^2] - (\mu_x + \mu_y)^2$$

$$var[X-Y] = E[X^2] - 2E[XY] + E[Y^2] - \mu_x^2 - 2\mu_x\mu_y - \mu_y^2$$

$$var[X-Y] = (E[X^2] - \mu_x^2) + (E[Y^2] - \mu_y^2) - 2(E[XY] + \mu_x\mu_y)$$

$$var[X-Y] = var[X] + var[Y] - 2cov[X,Y]$$

3.2 Bayes rule for quality control

Given
P = Test Positive, D = Defective

$$p(P|D) = .95$$

$$p(P^c|D^c) = .95$$

$$p(D) = 1e^{-5}$$

$$p(D|P) = ?$$

$$p(P) = p(D) * p(P|D) + (p(D^c) + (\,1 - p(P|D)\,)$$

$$p(P) = (1e^{-5} * .95) + ((1 - e^{-5}) * .05) = .050009$$

$$p(D|P) = \frac{p(D) * p(P|D)}{p(P)} = 1.899e^{-4}$$

The chances that a device is defective given that the test comes out positive is 1.899exp(-4).

The amount of good devices that are thrown away because they tested to be defective is calculated by using the following probabilities from the probability tree above:

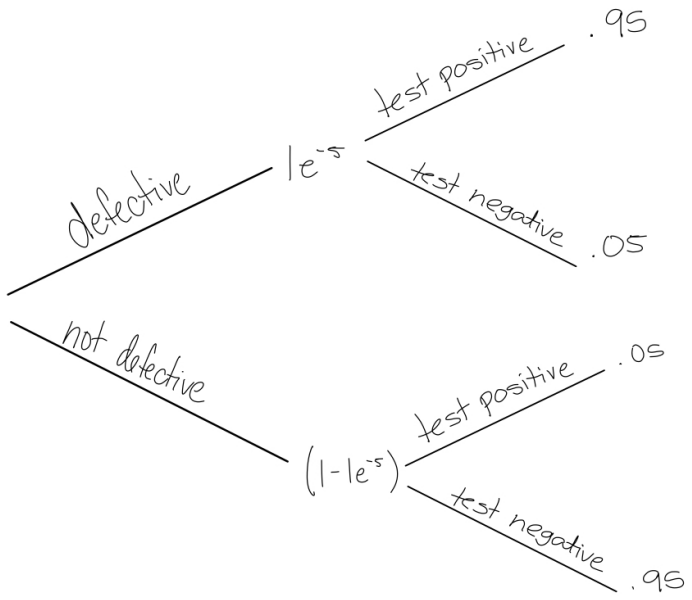

Figure 8: Probability Tree

$$p(P \cap D^c) * (amount\,of\,devices) = [(1 - 1e^{-5}) * .05] * 10e^6 = 499,955$$

Similarly, the amount of defective devices that are shipped is calculated with the following probabilities:

$$p(D \cap P^c) * (amount\,of\,devices) = [1e^{-5} * .05] * 10e^6 = 5$$

3.3 k-nearest neighbors

3.3.1 a

The average 0-1 prediction error in the training data when K = 1 will be 0. This is due to when K = 1 we are just considering the point itself, therefore

the prediction will always be correct.

However, when K increases, we are going to get higher and higher error as we will be getting the mode of the labels for K neighbours. As more neighbours considered, more inaccurate will the prediction be.

3.3.2 b

The average 0-1 prediction error on the held-out half (validation) will most probably be high when K values are low. This is due an over-fitting of the model to the training data which does not let the model generalize well in the test data.

On the opposite side, when K has higher values the average prediction error will most likely be high as well. When considering too many neighbours the proximity effect we are looking for in the KNN algorithm is lost. This could be consider an under-fitted model. Therefore, when training our KNN model we should aim for neither a too small neither a too big K, and the definition of what is too big or what is too small will depend on the nature of the data set. There are no rules.

A good way to find a K which minimizes the average prediction error is to perform cross-validation.

3.3.3 c

Performing K-folds cross validation is computationally expensive as the model is being trained k times. Nonetheless, cross validation will most likely prevent over-fitting. For each fold and each K value an accuracy is obtained. The final K will be the one that maximises the average accuracy obtained across the folds.

There is a trade-off between the computational requirements and the desired accuracy. In order to chose how many folds to use, we must consider how big our data set is, how much time and computational power do we have at our disposal, and how important it is to correctly predicting all our examples.

3.3.4 d

To avoid this caveat, a weight value could be introduced that so the closest neighbours would have a higher importance when predicting the new data point label.

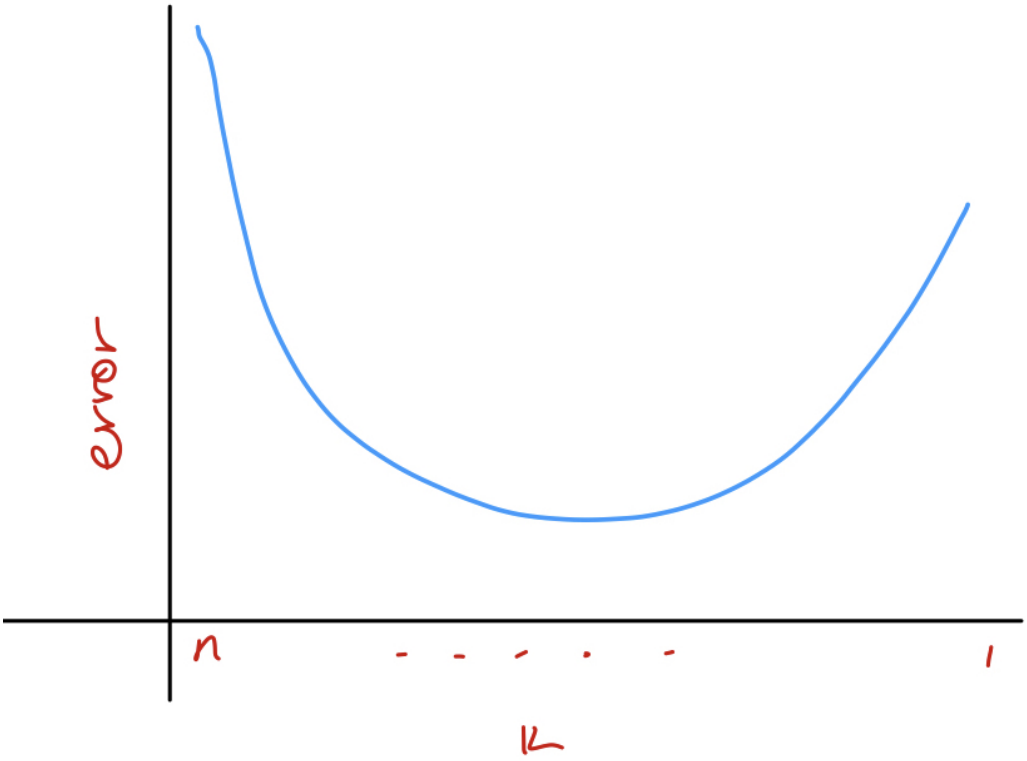

Figure 9: Prediction error depending on K

In other words, instead of using the mode of the K closer neighbours as a prediction, it would be interesting to use a weighted mode.

3.3.5 e

KNN is time consuming and computationally expensive. This is because KNN is a lazy learning method, meaning that there is no model. Every time a KNN is predicting a new point, it has to go through all the training set and compute all the distances to determine which points are the closest. When the data set has high dimensions the time to compute will increase exponentially.

Another reason why not to use KNN with a high dimensional input is because of the famous *Curse of Dimensionality*. The model should be generic and able to contemplate anomalies outside cases but with too many dimen-

sions it has a hard time doing so.