Topic 2: Supervised Learning and Classification

Seve Martinez

Grand Canyon University

Abstract

The MNIST dataset is a common tool to leverage and test classification algorithms. One of particular interest is the k-Nearest Neighbors algorithm, which uses Euclidean distance in order to determine similar points and a voting system to classify the point in question. This paper uses this dataset and the SciKit-Learn Python library's KneighborsClassifier class to build a model that can successfully classify handwritten digits with as high as 96.6% accuracy.

Keywords: k-nearest neighbors, sklearn, mnist, euclidean distance, confusion matrix

Topic 2 Technical Report

Handwritten digit identification is particularly challenging due to the unique way in which many people write text. Standard practices of handwriting teaching have been lost to the age of computing. However, in many scenarios there is a use for this type of work. Some examples include form recognition, translation software (Google Lens) or postal addresses.

The k-nearest neighbors algorithm is a simple and useful method to classify this type of data. It's attractiveness lies in it's simplicity. K is a parameter that sets the number of neighbors to consider in the voting process, and each neighbor is determined by Euclidean distance.

The Dataset

MNIST Handwritten Image Data

MNIST a series of 28x28 images that contain handwritten digits in grayscale. There are 60,000 training images and 10,000 testing images. The particular split of the data for this report was not heavily considered as best practice and common usage split the data in this fashion. In fact, this split is so ubiquitous that the data is even presented in this manner. There is a good distribution of each digit within the data so no class is too over- or under-represented.

Had there been fewer data points then perhaps a more generous testing split would have been considered.

This data was ideal as it had been converted to various color intensities varying from 0-255 in a 784-column table. This table was then converted to csv for ease of manipulation in a Pandas DataFrame.

Testing Methodologies

ROC-AUC Curve

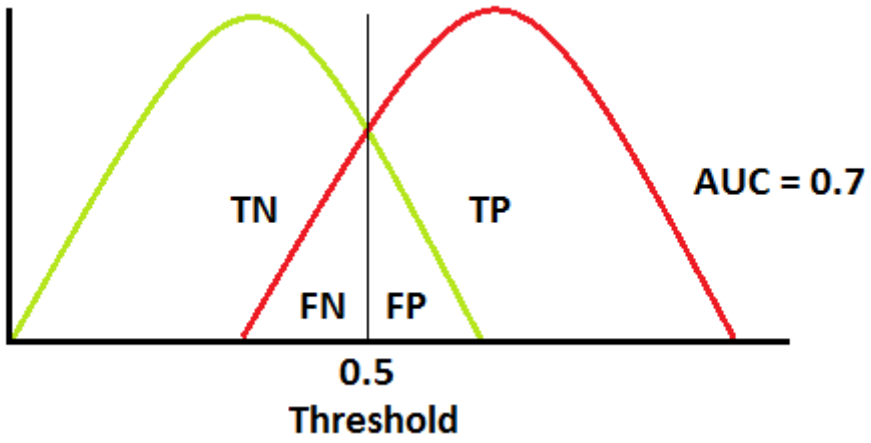
The Receiver Operating Characteristics (ROC) Area Under the Curve (AUC) is used to determine how well the model is able to distinguish between classes. The scale is from 0 to 1 with 1 being the best and 0 being completely unable to discern. Generally, the higher the number the better.

This is calculated using the true positive and false positive rate. True positives are values that were correctly identified as positive and false positives are negative values that were incorrectly classified as positive. These two values are calculated as follows:

True Positive Rate: $$\frac{TP}{TP + FN}$$ (Equation 1)

False Positive Rate: $$\frac{TN}{TN + FP}$$ (Equation 2)

Narkhede provides us a nice graphic:



Confusion Matrix

The confusion matrix (CM) is a summary of prediction results from a classification problem. It is broken down by class into a table which shows where correct and incorrect predictions are made. By this, it shows where the model was "confused" in classifiying a particular value.

The confusion matrix is ideal for multi-class problems because it divides predictions by class so that we may see which class a particular value was labeled. Traditional accuracy calculations do not provide the detail necessary to show where the model is having problems.

If we were to simply use an accuracy calculation of:

$$accuracy = totalcorrectpredictions/totalpredictionsmade * 100 \quad \text{(Equation 3)}$$

This would not show us what was misclassified, making it more difficult to accurately tune the model. For this particular problem, the CM was leveraged to show which value a particular digit was labeled. There were ten digits, therefore the grid is 10x10

The diagonal shows the correct values:

```
[[ 975,    1,    0,    0,    0,    0,    2,    1,    1,    0],
       [   0, 1128,    3,    1,    1,    0,    0,    2,    0,    0],
```

```
[  19,    1,  990,    0,    0,    0,    0,   10,   12,    0],
[   6,    0,    3,  964,    1,    7,    0,    6,   15,    8],
[   8,    5,    0,    1,  943,    0,    6,    2,    3,   14],
[  10,    4,    0,   28,    1,  833,    8,    0,    5,    3],
[   5,    5,    1,    0,    1,    2,  943,    0,    1,    0],
[   5,    7,    8,    0,    4,    0,    0,  983,    0,   20],
[   9,    4,    2,    6,    4,    3,    2,    6,  934,    4],
[  16,    5,    4,    4,   17,    2,    1,    9,    5,  946]])
```

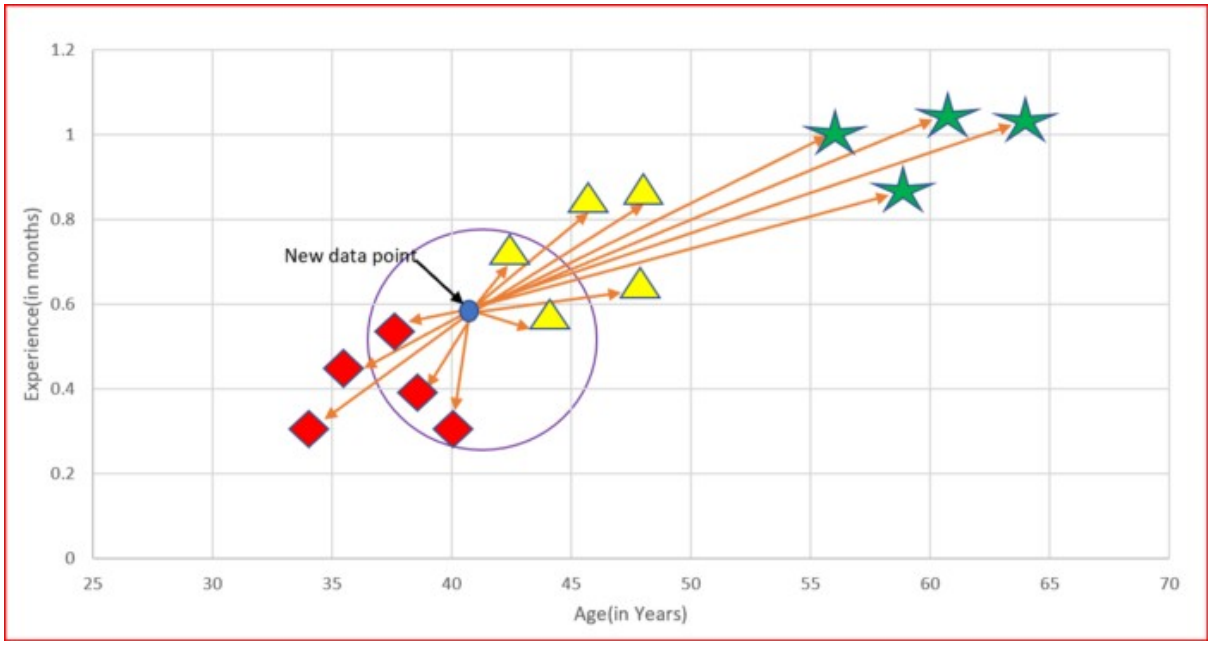
The Algorithm

K-NN Classifier

The k-NN classifier is often lauded for its simplicity. The main function uses Euclidean distance, defined as follows:

$$\mathrm{d(p,q)} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \qquad \text{(Equation 4)}$$

Essentially the magnitude of the vector is calculated by subtracting a training point from a testing point. Once this is done, then a voting system of nearby points allows us to determine what the value should be. This is where the “k” in “K-NN” comes in: K represents the number of neighbors used to vote. This is an arbitrary value that is best found by trial and error. The graphic below shows an example.



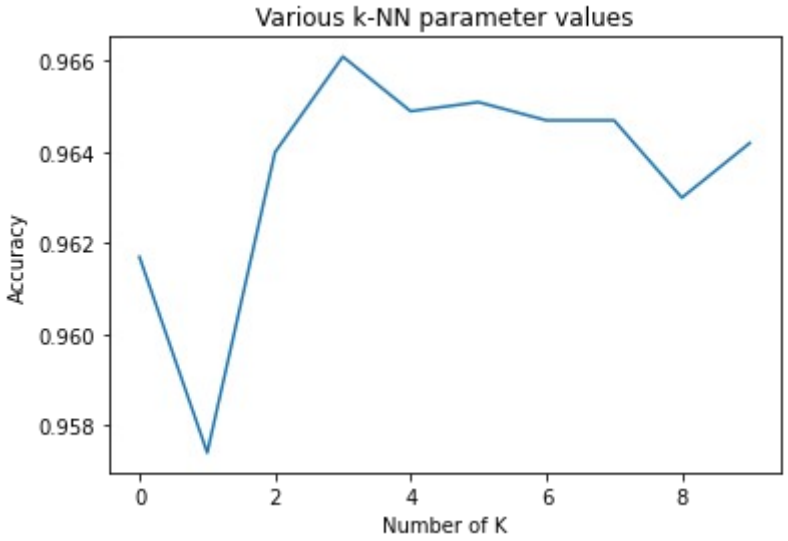
For this research, a model was built using the sklearn class KNeighborsClassifier. This has one main parameter, n_neighbors that represents our k value. In order to find the best k, a loop was built to vary k and provide the results in a list. That list was then plotted to see which value of k brought about the best accuracy. N_jobs was also set to 14 to leverage the horsepower available on a 16 core machine. Two were left out to prevent machine lockup.

Findings

The most accurate model resulted in a 96.6% score and determined that the digit 1 was the most popular with 1160 values. The most ideal k was 3 neighbors. The breakdown of the other classified digits is presented below.

```
2: 1011,
1: 1160,
0: 1053,
 4: 972,
 9: 995,
 5: 847,
 6: 962,
7: 1019,
3: 1004,
 8: 976
```

The loop described earlier to determine the best k was plotted and each value of the k was put up as a percentage. As the neighbors increased, the model appeared to worsen. This makes sense since the larger the number, the more dissimilar points would be in the voting process.



References

References:

Brownlee, J. (2020). What is a Confusion Matrix in Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/confusion-matrix-machine-learning/

Narkhede, S. (2018). Understanding Confusion Matrix. Towards Data Science. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Narkhede, S. (2018). Understanding AUC-ROC Curve. Towards Data Science. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Khandelwal, R. (2018). K-Nearest Neighbors (KNN). Towards Data Science. https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7

Gopal. Applied Machine Learning. https://www.gcumedia.com/digital-resources/mcgraw-hill/2019/applied-machine-learning_1e.php

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.