**CSG2341.3 Assignment 2.1**

**APPROACH**

The k-Nearest Neighbours algorithm is a simple supervised machine learning algorithm that can be used to make classification or regression predictions, based on the assumption that similar data points are likely to be grouped together in a dataset. The algorithm works by finding the *k* closest data points to a given value, i.e. its *nearest neighbours*, and simply assigning it whichever class represents the majority among these neighbours (IBM, n.d.). For example, if the majority of a data point X’s nearest neighbours are of class A, then a k-NN algorithm will predict that point X is also of class A. In cases where more than two classes are present but none has an outright majority, the algorithm will predict whichever class is represented more than the others, known as the plurality.

To determine the ‘closeness’ of potentially abstract data points, a suitable distance measure for the nature of the data is chosen. Numerical data is typically plotted as Cartesian coordinates, as with the Manhattan Distance metric, which measures the distance between the x and y coordinates of two points along axes at right angles (National Institute of Standards and Technology, 2019). The Hamming Distance metric is used for calculating distance between data points of plain text, comparing the number of similar and different letters between two words of the same length to derive a measure of distance (Analytics Vidhya, 2023). This paper will use Euclidean Distance to determine the nearest neighbours of numerical datapoints. Euclidean Distance can be visualised by plotting points of numerical data as Cartesian coordinates and measuring the length of a line segment joining two given points.

In this paper I will implement the k-NN algorithm to perform classification predictions, as part of a Python program that will accurately identify handwritten numbers provided in a dataset of small grayscale digital images (Alpaydin & Kaynak, 1998). The algorithm will be trained on a subset of already-identified images before being tested on the remaining unidentified images in the dataset. The algorithm results will displayed in a series of figures, outlining the accuracy of its classification predictions and how the algorithm can be optimised with different values of *k*.

**IMPLEMENTATION** (https://blog.devgenius.io/understanding-handwritten-digit-recognition-using-k-nearest-neighbors-knn-da677e87c8ac)

* Load dataset
* Plot distribution of classes\*
* Plot average images\*
* Plot distribution of pixel intensities\*
* Prepare the dataset for kNN (normalise grayscale values)
* Split data into training set and testing set (20%)
* Define parameters (k, euclidean distance)
* Perform grid search to determine optimal parameters\*
* Train model
* Test model
* Evaluate Results

(flowchart here)

**PARAMETERS**

The k-NN machine learning algorithm requires only two parameters: the value *k*, which defines the number of neighbouring data points that will be used to classify the target data point, and a metric by which the distance between data points can be measured. This implementation of the algorithm uses Euclidean Distance as its distance metric, and demonstrates how varying values of *k* can effect the prediction accuracy of the algorithm. *k* values used will be exclusively odd, whole numbers, in order to eliminate ambiguity in cases where two neighbouring data points may be the same distance apart. \*\*\* TODO: Only 3, 5, 7, etc \*\*\*

**LANGUAGE AND TOOLS USED**

This implementation is written in Python, taking advantage of the scikit-learn tool to streamline the process of importing the dataset and preprocessing data into training and testing subsets (https://scikit-learn.org). Matplotlib, a tool for creating data visualisations, is used to plot graphs and tables of the algorithm’s results (https://matplotlib.org/).

**Performance Evaluation Section (individual):**

**DATASET**

The MNIST, or Modified National Institute of Standards and Technology, dataset, is made up of 70,000 images of handwritten digits (LeCun, et al. 1998). Each is a 28x28 pixel grayscale image of a single number from 0-9. 60,000 images are labelled and used for training the algorithm, with the remaining unlabeled images used for testing.

**FINDINGS**

Optimal parameters

Accuracy results

Confusion Matrix

**Performance Comparison Section (joint)**

- Test algorithm with Iris dataset

- Demonstrate findings with tables/figures

**REFERENCES:**

**https://www.ibm.com/topics/knn**

**https://xlinux.nist.gov/dads/HTML/manhattanDistance.html**

[**https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/**](https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/)

**https://ieeexplore.ieee.org/document/726791**