COMP9517

Lab 3, T2 2019 (02/07/2019)

The goal of this lab is to become familiar with the training/testing process for pattern recognition (week 4 lectures). A simple k-nearest neighbour (kNN) algorithm is to be developed. Background information is provided AFTER all the questions.

Code and results should be submitted via WebCMS3 for marking (1 mark). Submission is due at 23:59:59 on July 2nd, 2019. Submission instruction will be posted prior to the lab.

Preliminaries

The following experiments will be based on a provided dataset of handwritten digits (500 samples of each digit 0 to 9). This dataset was derived from the 'digits.png' file included with OpenCV installations. This dataset is designed to test classification algorithms.

Some Python libraries you might find useful for this lab include scikit-learn, numpy and opencv. For reading images from disk, the os and glob libraries will be useful.

It is recommended that you use the "KNeighborsClassifier" class from scikit-learn instead of the kNN classifier from opencv, because scikit-learn functions give you more control and you can set different parameters to optimise the kNN classifier.

The "metrics" module from scikit-learn is useful for generating the confusion matrix.

You are free to use other programming languages that you prefer, such as Matlab.

Lab Tasks (1 mark)

Develop a program to perform handwritten digit recognition using the kNN classifier. The program should contain the following steps:

- import packages
- read the dataset (images and class labels)
- split images dataset into 80% training and 20% test sets
- initialize kNN model (use k=5)
- fit the KNN model using the training data (i.e. construct a search tree)
- perform handwritten digit recognition using the test data
- evaluate the recognition performance by calculating accuracy, confusion matrix, and precision and recall for each digit class

Submit the results (accuracy, confusion matrix, per-class precision and recall) in a pdf file and code for marking.

Extra Tasks (not assessed)

Experiment with different k values and different parameters in KNeighborsClassifier from scikit-learn, and see their effects on the recognition accuracy and efficiency.

Refer to the scikit-learn documentation for available parameters:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Background

K - Nearest Neighbours

The KNN algorithm is very simple and very effective. The model representation for KNN is the entire training dataset. Predictions are made for a new data point by searching through the entire training set for the K most similar instances (the neighbours) and summarizing the output variable for those K instances. For regression problems, this might be the mean output variable, for classification problems this might be the mode (or most common) class value. The trick is in how to determine the similarity between the data instances.

Similarity

In order to make predictions we need to calculate the similarity between any two given data instances. This is needed so that we can locate the k most similar data instances in the training dataset for a given member of the test dataset and in turn make a prediction.

Given that all four flower measurements are numeric and have the same units, we can directly use the Euclidean distance measure. This is defined as the square root of the sum of the squared differences between the two arrays of numbers (read that again a few times and let it sink in).

References:

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_index.html