COMP9517: Computer Vision
2020 T2 Lab 4 Specification
Maximum Marks Achievable: 2.5

This lab is **worth 2.5% of the total course marks**.

The lab files should be submitted online.
Instructions for submission will be posted closer to the deadline.
Deadline for submission is Week 7, Monday 13 July 2020, 23:59:59.

Objective: This lab revisits important concepts covered in the lectures of Week 5 and aims to make you familiar with implementing specific algorithms.

Materials: You are required to use OpenCV 3+ with Python 3+. Jupyter notebook files are preferred for submitting your code.

Submission: The task is assessable **after the lab**. Submit your code and results in a zip file via WebCMS3 by the above deadline.

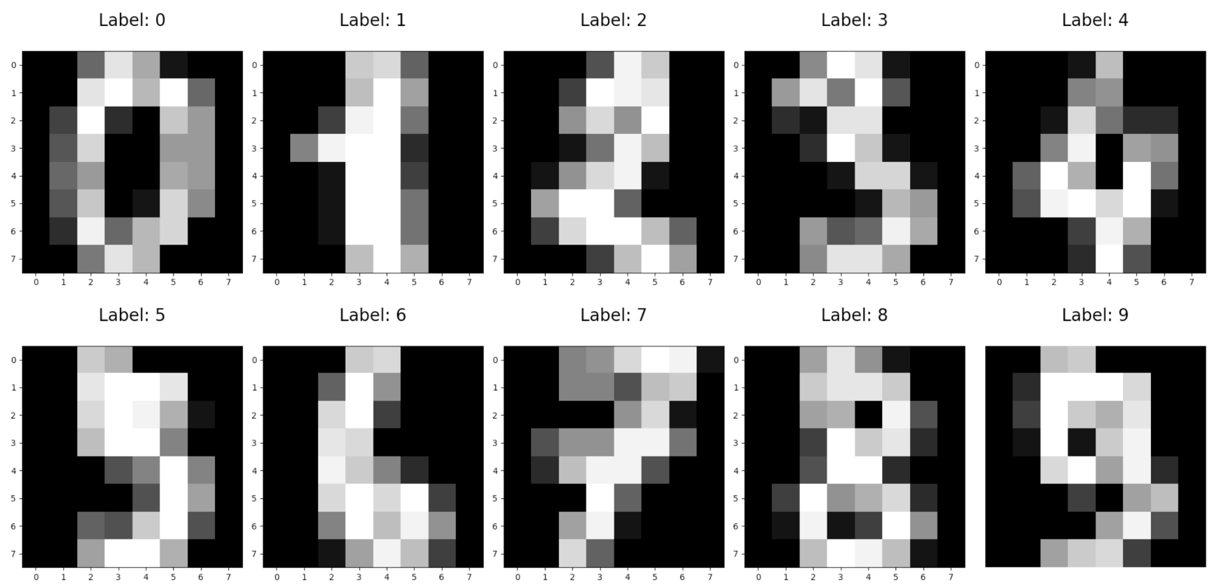
Pattern Recognition

To goal of this lab is to implement a K-Nearest Neighbours (KNN) classifier, a Stochastic Gradient Descent (SGD) classifier, and a Decision Tree (DT) classifier. **Background information on these classifiers is provided at the end of this document**.

The experiments in this lab will be based on scikit-learn's **digits** data set which was designed to test classification algorithms. This data set contains 1797 low-resolution images (8×8 pixels) of digits ranging from 0 to 9, and the true digit value (also called the label) for each image is also given (see examples on the next page).

We will predominantly be using scikit-learn for this lab, so make sure you have downloaded it. The following scikit-learn libraries will need to be imported:

```
from sklearn import metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```



Sample of the first 10 training images and their corresponding labels.

Task (2.5 marks): Perform image classification on the digits data set.

Develop a program to perform digit recognition. Classify the images from the digits data set using the three classifiers mentioned above and compare the classification results. The program should contain the following steps:

Set Up

Step 1. Import relevant packages (most listed above).

Step 2. Load the images using sklearn's `load_digits()`.

Optional: Familiarize yourself with the dataset. For example, find out how many images and labels there are, the size of each image, and display some of the images and their labels. The following code will plot the first entry (digit 0):

```
plt.imshow(np.reshape(digits.data[0], (8, 8)), cmap='gray')
plt.title('Label: %i\n' % digits.target[0], fontsize=25)
```

Step 3. Split the images using sklearn's `train_test_split()` with a test size anywhere from 20% to 30% (inclusive).

Classification

For each of the classifiers (`KNeighborsClassifier`, `SGDClassifier`, and `DecisionTreeClassifier`) perform the following steps:

Step 4. Initialize the classifier model.

Step 5. Fit the model to the training data.

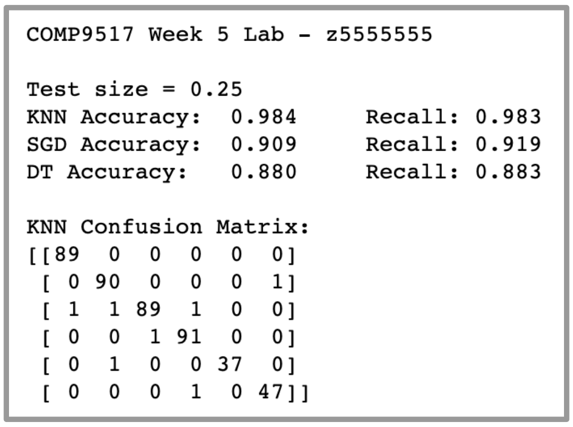
Step 6. Use the trained/fitted model to evaluate the test data.

Evaluation

Step 7. For each of the three classifiers, evaluate the digit classification performance by calculating the **accuracy**, the **recall**, and the **confusion matrix**.

Experiment with the number of neighbours used in the KNN classifier in an attempt to find the best number for this data set. You can adjust the number of neighbours with the `n_neighbours` parameter (the default value is 5).

Print the **accuracy and recall of all three classifiers** and the **confusion matrix of the best-performing classifier**. Submit a screenshot for marking (see the example below for the case of just a 6-class model). Also submit your code and include a brief justification for the chosen parameter settings for KNN.

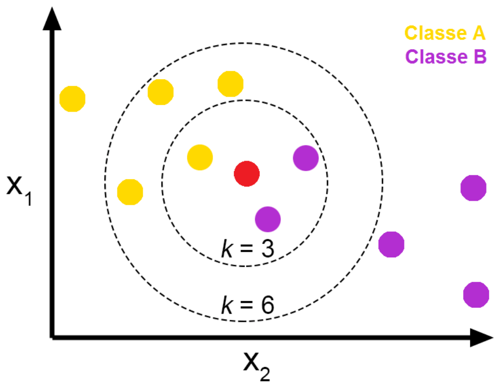


```
COMP9517 Week 5 Lab - z5555555

Test size = 0.25
KNN Accuracy:  0.984     Recall: 0.983
SGD Accuracy:  0.909     Recall: 0.919
DT Accuracy:   0.880     Recall: 0.883

KNN Confusion Matrix:
[[89  0  0  0  0  0]
 [ 0 90  0  0  0  1]
 [ 1  1 89  1  0  0]
 [ 0  0  1 91  0  0]
 [ 0  1  0  0 37  0]
 [ 0  0  0  1  0 47]]
```

Background Information

K-Nearest Neighbours (KNN)

The KNN algorithm is very simple and very effective. The model representation for KNN is the entire training data set. 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.

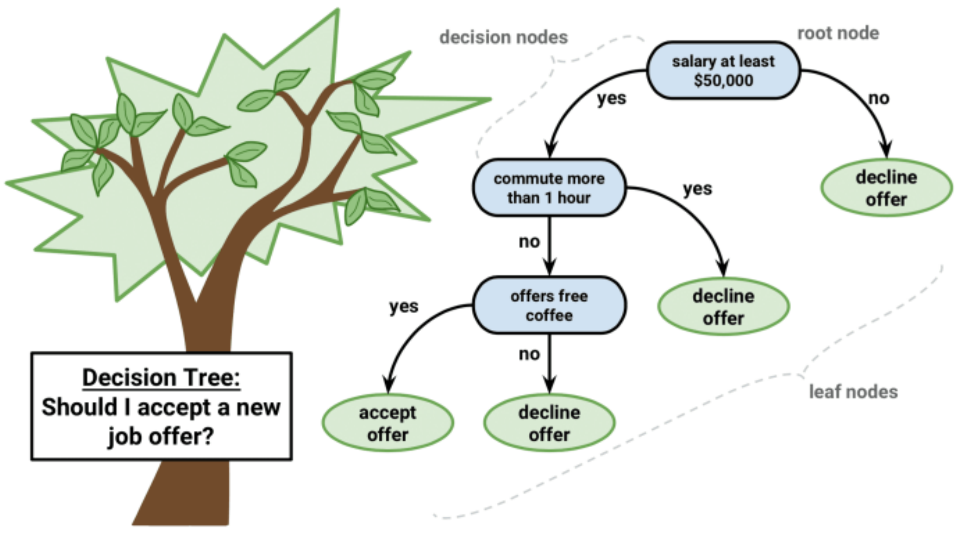


A 2-class KNN example with 3 and 6 neighbours (from Towards Data Science).

Similarity: To make predictions we need to calculate the similarity between any two data instances. This way we can locate the K most similar data instances in the training data set for a given member of the test data set and in turn make a prediction. For a numeric data set, 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.

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

Decision Tree (DT)
See https://en.wikipedia.org/wiki/Decision_tree_learning for more information.



The algorithm for constructing a decision tree is as follows:
1. Select a feature to place at the node (the first one is the root).
2. Make one branch for each possible value.
3. For each branch node, repeat Steps 1 and 2.
4. If all instances at a node have the same classification, stop developing that part of the tree.

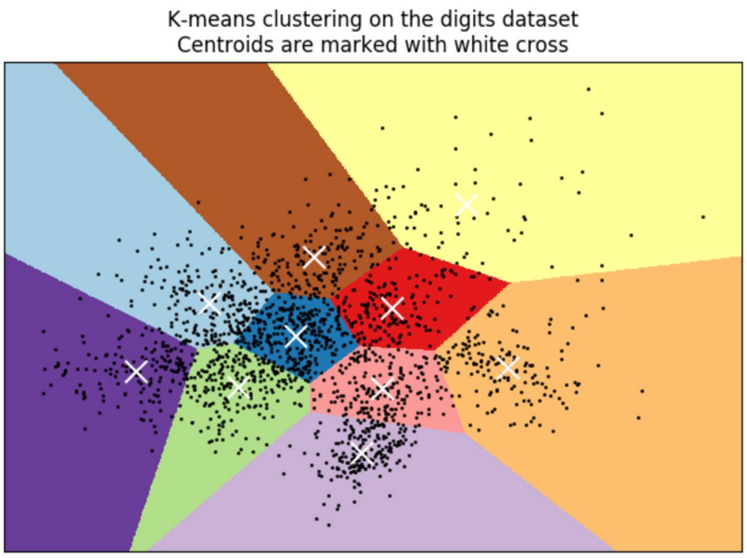
How to determine which feature to split on in Step 1? One way is to use measures from information theory such as **Entropy** or **Information Gain** as explained in the lecture.

Stochastic Gradient Descent (SGD)

See https://scikit-learn.org/stable/modules/sgd.html for more information.

Experiment with Different Classifiers

See https://scikit-learn.org/stable/modules/multiclass.html for more information. There are many more models to experiment with. Here is an example of a clustering model:



References

Wikipedia: K-Nearest Neighbors Algorithm
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

OpenCV-Python Tutorials: K-Nearest Neighbour https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_index.html

Towards Data Science: KNN (K-Nearest Neighbors)
https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d

SciKit-Learn: sklearn.neighbors.KNeighborsClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

SciKit-Learn: Demo of K-Means Clustering on the Handwritten Digits Data
https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

---

Released: 2 July 2020