COMP9517: Computer Vision

2023 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 10 July 2023, 18:00:00.

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 Python 3+, TensorFlow2, and Scikit-learn. The dataset used in this lab is the Dogs versus Cats dataset available from the Kaggle website:
https://www.kaggle.com/competitions/dogs-vs-cats/data
The dataset consists of a training set of 25,000 images and a testing set of 12,500 images. The images are in RGB-colour format and may have different sizes, but each image belongs to only one of two classes: Dogs versus Cats. In this lab, we will use only the training set, as the true class labels of the testing set are not available from the website.

Submission: The task is assessable after the lab. Submit your source code as a Jupyter notebook (.ipynb) which includes all output (see coding requirements below) by the above deadline. The submission link will be announced in due time.

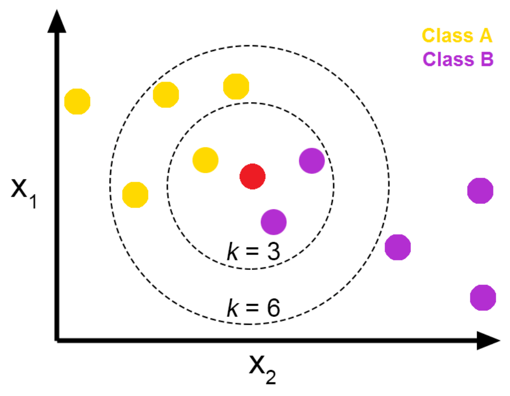
Pattern Recognition

The goal of this lab is to implement and compare a K-Nearest Neighbours (KNN) classifier, a Decision Tree (DT) classifier, and a Stochastic Gradient Descent (SGD) classifier. Below we provide a brief overview of these classifiers before specifying the task for this lab.

K-Nearest Neighbours (KNN) Algorithm

The KNN algorithm is very simple and 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.



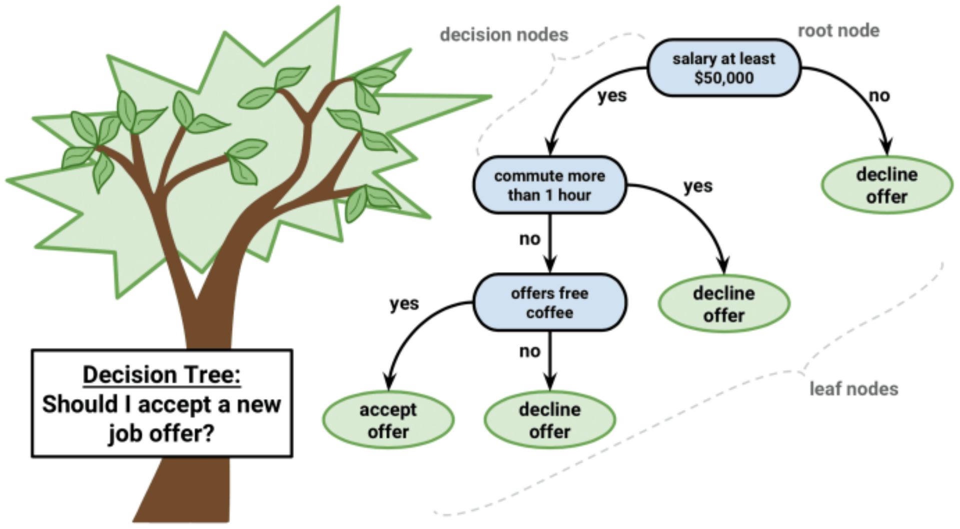
A 2-class KNN example with 3 and 6 neighbours (from Towards Data Science).

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 dataset for a given member of the testing dataset and in turn make a prediction. For a numeric dataset, 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) Algorithm

See also https://en.wikipedia.org/wiki/Decision_tree_learning for more information.



The algorithm for constructing a decision tree is as follows:

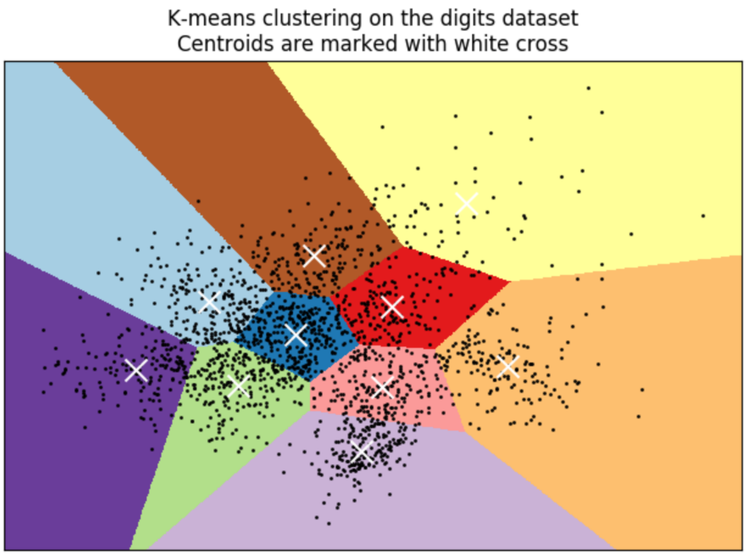
1. Select a feature to place at the node (the first one is the root).
2. Make one branch for each possible value.

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) Algorithm
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:



Task (2.5 marks): Perform image classification on the given dataset.

Develop a program to perform pattern recognition for the Dogs versus Cats dataset. Classify the images using the three classifiers KNN, DT, and SGD, and compare the classification results. The program should contain the following steps:

Setup

Step 1: Import relevant packages

We will predominantly be using **Scikit-learn** for this lab, so make sure you have correctly installed the Scikit-learn library before moving to the next steps. You can check the following link to know more about the library and ways to install it:
https://scikit-learn.org/stable/index.html

Check the following links on how to import KNN, DT, and SGD classifiers:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

Step 2: Load the dataset

After successfully downloading the Dogs versus Cats dataset, familiarize yourself with it. Specifically, you can check how many images there are in the entire dataset and in each class, and what is the size of each image. Also, display some images of each class. You are encouraged to experiment with preprocessing of the images to improve your classification results.

Step 3: Take a subset of the dataset

As you can see, there are 25,000 images (RGB-colour format) in the training set, and the correct label (Dog versus Cat) follows from the file name. There is also a testing set, consisting of another 12,500 images, but the class labels are not given, so we will not use that set. Instead, we split the training set into a subset used for the actual training, and a subset for testing.

To reduce computations, we can work on a subset of the complete training dataset. Initially, split this dataset into 10,000 images for training and 5,000 images for testing. After completing all the steps below, double the number of training images to 20,000 while keeping the number of testing images fixed at 5,000, and repeat the experiment to see if it makes a difference in classification performance. To avoid class imbalance in the training set, which may negatively impact performance, make sure to sample an approximately equal number of images from the Dogs class and the Cats class.

See also the Scikit-learn built-in function `train_test_split()` which can automatically shuffle the dataset and help you to split the data:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Step 4: Perform necessary reshaping of the data for the classifiers

Once you obtained a subset of the dataset to work with, you need to reshape the training and testing data in order to apply the machine learning classifiers. Each image in the Dogs versus Cats dataset should have same size, so you need to resize it.

Classification

Perform the following steps for each of the classifiers:

Step 5: Initialise the classifier model

After you import each classifier from the Scikit-learn library, you need to instantiate (initialise) the model (classifier) object. It is important to read the documentation to find out various parameters that can be configured when initialising classifier models.

Step 6: Fit the model to the training data

The Scikit-learn library has a fitting method to learn from data. Using the `fit()` method, train each classifier by passing training data and training labels as parameters.

Step 7: Evaluate the trained model on the testing data

After you have trained a classifier (also called a model), you can use it to make predictions on the testing data. Using the `predict()` method provided by the Scikit-learn library.

Evaluation

Step 8: Report the performance of each classifier

To quantify the performance of the trained classifiers, use standard classification metrics such as **accuracy**, **precision**, **recall**, and the **F1-score**. To summarise the correct and incorrect results for each class, use the **confusion matrix**. The Scikit-learn library provides built-in methods to automatically calculate these measures by comparing the predicted labels with the provided ground-truth labels. Click on the following link to find these methods and import them:
https://scikit-learn.org/stable/modules/model_evaluation.html

For each classifier, show the values of all of the above-mentioned standard classification metrics and the confusion matrix in your Jupyter notebook. Also compare the accuracy of your classifier with the scores seen on the leaderboard of the Dogs versus Cats competition and give an explanation why your results are better or worse.
https://www.kaggle.com/competitions/dogs-vs-cats/leaderboard

Coding Requirements

In your Jupyter notebook, all cells should have been executed so that the tutor/marker does not need to execute them again to see the results.

References

Dogs Versus Cats Dataset:
https://www.kaggle.com/competitions/dogs-vs-cats/data

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

OpenCV-Python Tutorials: K-Nearest Neighbour https://opencv24-python-tutorials.readthedocs.io/en/stable/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: 30 June 2023