COMP9517: Computer Vision

2021 T3 Lab 3 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 5, Wednesday 13 October 2021, 23:59:59.

Objective: This lab revisits important concepts covered in the lectures of Week 4 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 Fashion-MNIST dataset available through TensorFlow2. Fashion-MNIST consists of a training set of 60,000 example images and a test set of 10,000 examples. Each example is a 28 x 28 pixel grayscale image associated with a label from 10 classes:
https://github.com/zalandoresearch/fashion-mnist

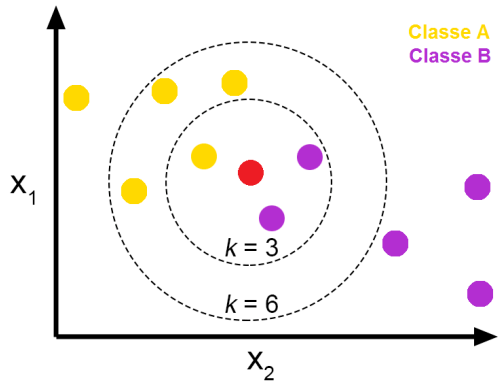
Submission: The task is assessable **after the lab**. Submit your source code as a Jupyter notebook (.ipynb) with all output included in a single zip file via Moodle by the above deadline. The submission link will be announced in due time.

Pattern Recognition

To 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 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.

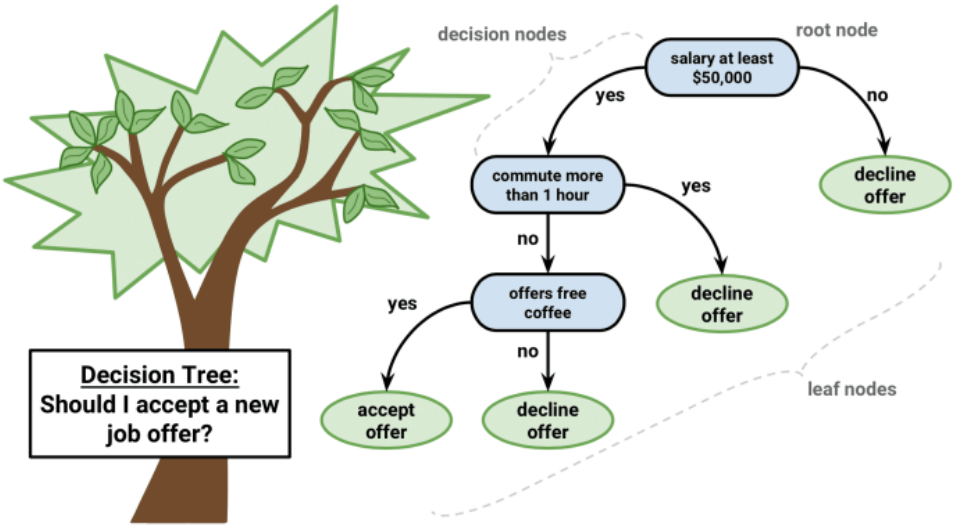


A 2-class KNN example with 3 and 6 neighbours (from Towards Data Science).

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 test 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.

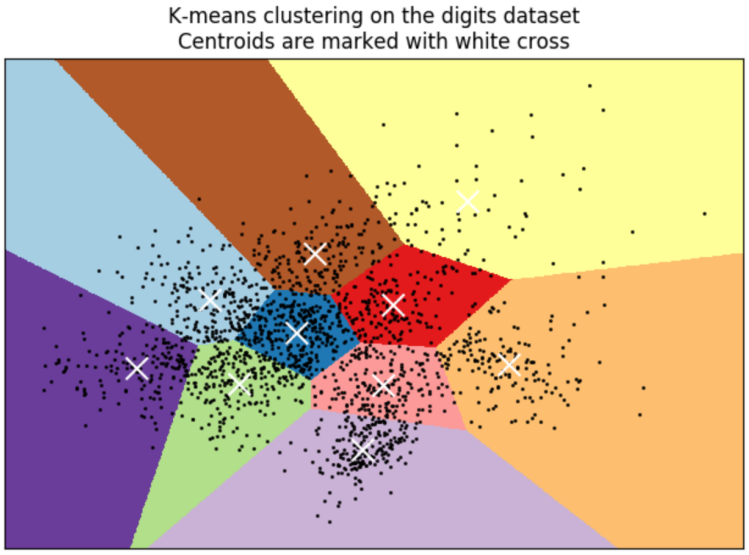


The algorithm for constructing a decision tree is as follows:
1. Select a feature to place at the node (the first one is the root).
2. Make one branch for each possible value.
3. For each branch node, repeat Steps 1 and 2.
4. If all instances at a node have the same classification, stop developing that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

Develop a program to perform pattern recognition for the Fashion-MNIST dataset. Classify the images using the three classifiers KNN (k = 3), DT, SGD (max_iter = 250), 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 Fashion-MNIST dataset

If you check the README file at https://github.com/zalandoresearch/fashion-mnist, you will find that many machine learning libraries and deep learning frameworks have included Fashion-MNIST as a built-in dataset. This means we do not need to manually download the Fashion-MNIST dataset, rather we can use the API, which can automatically download the dataset for us.

Keras, a deep learning API written in Python running on top of the deep learning framework TensorFlow, provides capability to automatically download the Fashion-MNIST dataset. Check the following link to use the Keras API for downloading the Fashion-MNIST dataset:
https://keras.io/api/datasets/fashion_mnist/

After successfully downloading the Fashion-MNIST dataset, familiarize yourself with it. Specifically, you can check how many images and labels are there in the entire dataset, what is the size of each image, how many classes there are. Also, display some images of each label.

Step 3: Take a subset of the data set (2,000 for training and 500 for testing)

As you can see, there are 60,000 samples (grayscale images) in the training set and 10,000 samples in the testing set. To reduce computation, we can work on a subset of the complete dataset. Take 2,000 samples for training and 500 samples for testing. The Fashion-MNIST dataset you imported using TensorFlow is already split into training and testing sets. In case you got the complete dataset, you can use the scikit-learn in-built function `train_test_split()` which automatically shuffles the dataset and helps 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 got a subset of dataset to work with, you need to reshape the training and testing data in order to apply machine learning classifiers. Each image in the Fashion-MNIST dataset is 28 x 28 pixels, so you need to reshape it.

Classification

Perform the following steps for each of the classifiers:

Step 5: Initialise the classifier model

After you import each classifier from scikit-learn library, you need to instantiate (or 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: Use the trained/fitted model to evaluate the test data

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

Evaluation

Step 8: Reporting performance of each classifier

To check how well each of the trained classifier is performing, you can report standard classification metrics such as **accuracy**, **precision**, **recall**, and the **F1-score**. You can also see the errors for each class using the **confusion matrix**. The scikit-learn library provides in-built methods to automatically calculate these measures by comparing ground-truth labels and the predicted 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 for all of the above-mentioned standard metrics and the confusion matrix in your Jupyter notebook.

Coding Requirements and Suggestions

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

Zalando Research Fashion-MNIST Data Set:
https://github.com/zalandoresearch/fashion-mnist

Fashion-MNIST Benchmarking Paper:
https://arxiv.org/pdf/1708.07747.pdf

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: 5 October 2021