COMP9517

Lab 3, T1, 2021

The lab files should be submitted online.

Instructions for submission will be posted closer to the deadline.

Deadline for submission is Week 5, TUESDAY 16 MAR 2021, 16:59:59

(Note deadline on the Tuesday for Lab3, due to assignment deadline on the Monday)

Objectives: This lab presents a revision of important concepts from week 4 lectures. The questions require you to use scikit-learn and tensorflow2, software packages widely used in this field.

Materials: You are required to use Python 3, tensorflow2 and scikit-learn.

Submission: Assessment question is assessable **after the lab** and is **worth 2.5% of the total course marks**. Submit your code and results as a Jupyter notebook in a zip file via WebCMS3 by the deadline.

1 Pattern Recognition

The dataset used for this entire lab is the fashion-MNIST dataset available through tensorflow2. Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes (the details of the classes can be found in the linked website).

Assessment Question: ***Classical Machine Learning***

Develop a program to perform pattern recognition for the dataset mentioned above. Classify the 10 fashion items using the three classifiers (KNeighboursClassifier [k=3], SGDClassifier [max_iter=250] and DecisionTreeClassifier) and compare their results. The program should contain the following steps:

1. Import relevant packages.
2. Load the dataset from tensorflow.
3. Familiarize yourself with the dataset.
4. Take a subset of the dataset (2000 training, 500 test).
5. Perform necessary reshaping of the dataset for the aforementioned classifiers.
6. Initialize models.
7. Fit models to training data.

8. Use the trained models to evaluate the test data.
9. For each classifier evaluate the performance by calculating the accuracy, recall and generating the confusion matrix inside your jupyter notebook.
10. Submit your jupyter notebook which includes all the results specified.

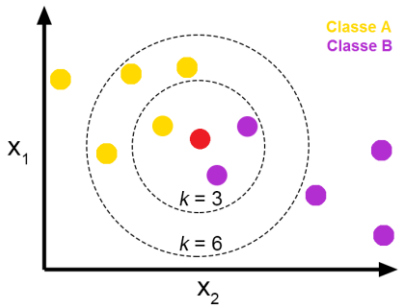
The confusion matrix should appear similar to the one provided below:

```
KNeighborsClassifier Confusion Matrix:
[[15  0  0  0  0  0 40  0  0  0]
 [ 3  1  2  0 23  0 23  0  0  0]
 [ 1  0 37  0  0  0 27  0  0  0]
 [ 6  0  4  3  2  0 31  0  0  0]
 [ 3  0 34  0  1  0 19  0  0  0]
 [ 0  0  0  0  0  0  3  0 28  8]
 [ 4  0 19  0  0  0 24  0  0  0]
 [ 0  0  0  0  0  0  0  0 44  3]
 [ 3  0  1  0  0  0 10  0 30  0]
 [ 0  0  0  0  0  0  3  0 17 28]]
```

Background Information

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.



2 Class KNN example with k = 3 and k = 6 neighbours (Towards Data Science)

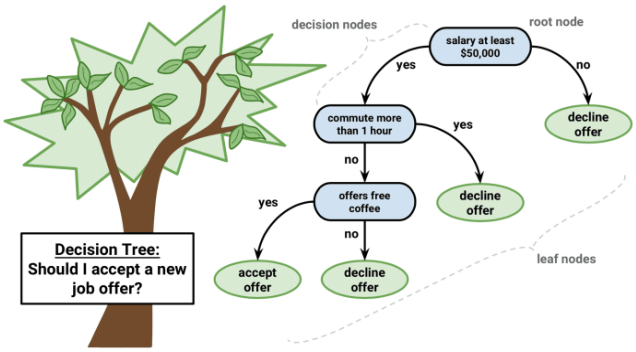
Similarity

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.

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.

Decision Tree Algorithm

(see also **https://en.wikipedia.org/wiki/Decision_tree_learning**)



Algorithm for Construction of Decision Tree

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 step 1 and 2
4. When all instances at a node have the same classification, stop developing that part of the tree

How to determine which feature to split on?

5. One way is to use measures from information theory***: Entropy*** and ***Information Gain***

Entropy

To construct optimal decision trees from training data, we need a definition of optimality. One simple criterion is ***entropy***, based on information theory. Entropy may be viewed as the average uncertainty of the information source.

Information Gain

Information gain is an entropy-based measure to evaluate features and produce optimal decision trees. When splitting, we use the feature with highest information gain to split on.

Stochastic Gradient Descent Classifier

(See https://scikit-learn.org/stable/modules/sgd.html)

Refer to the scikit-learn documentation for available parameters:

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
Experiment with different classifiers: https://scikit-learn.org/stable/modules/multiclass.html

2 REFERENCES

[1]. https://arxiv.org/pdf/1708.07747.pdf
[2]. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[3]. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_index.html
[4]. https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d
[5]. https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[6]. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html
[7]. https://keras.io/examples/mnist_cnn/