Middle East Technical University Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Spring 2019-2020

Homework 2: K-Nearest Neighbors Algorithm, Hierarchical Agglomerative Clustering, and Decision Trees

version 1.1

Due date: 19 April 2020, 23:59

1 Introduction

In this assignment, you will have the chance to get hands-on experience with k-nearest neighbors algorithm (k-NN), hierarchical agglomerative clustering (HAC), and decision trees. You will write your code in Python. You are only allowed to use drawing and plotting libraries provided that they do not implement k-NN, HAC, and decision tree algorithms. Like in the previous homework, you will write a report called **report.pdf**. The structure of the report is up to you.

2 K-Nearest Neighbors Algorithm (20 pts)

This section includes three tasks.

1. Implement the unweighted k-NN algorithm which uses Euclidean distance as the distance metric. Write your code in the function **kNN** in the file **task1.py**. The datasets are plain text files. Each line of the file is a (train/test) instance. Each instance consists of features and a label, separated by commas. While the features are floating-point numbers, the label is a text. When grading, we call the function with different **k** values and datasets.

2. In the first task, the function expected a **k** value as a parameter. In this task, you are going to find the best **k** value by using **K-fold cross validation**. Write your code in the function **find_best_k** in the file **task1.py**. When grading, we call the function with different **K** values and datasets.

3. The test set accuracy for the provided dataset can be improved without modifying the aforementioned functions. Address the following questions in your report.

   (a) What are the shortcomings of the Euclidean distance?

   (b) Why does the dataset trigger the shortcomings of the Euclidean distance?

   (c) How can the dataset be preprocessed so that the test set accuracy improves?

3 Hierarchical Agglomerative Clustering (40 pts)

In this section, you are given four unlabeled datasets, each of which has two features. You will apply the following linkage criteria on each dataset and visualize the results. The Euclidean norm is used in the criteria.

1. The Single Linkage Criterion:

$$\text{Single}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \min_{n,m} \|x_n - y_m\|$$

2. The Complete Linkage Criterion:

$$\text{Complete}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \max_{n,m} \|x_n - y_m\|$$

3. The Average Linkage Criterion:

$$\text{Average}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \|x_n - y_m\|$$

4. The Centroid Linkage Criterion:

$$\text{Centroid}(\{x_n\}_{n=1}^N, \{y_m\}_{m=1}^M) = \left\| \left(\frac{1}{N} \sum_{n=1}^{N} x_n \right) - \left(\frac{1}{M} \sum_{m=1}^{M} y_m \right) \right\|$$

The HAC algorithm runs until it merges all data points into one cluster. To see the effects of the criteria functions, you will stop the algorithm when **k** clusters remain. The **k** value is **2** for **dataset 1**, **dataset 2**, and **dataset 3**. For **dataset 4**, the **k** value is **4**.

For each dataset, you are expected to:

1. Cluster the data points until k clusters remain. Then, plot and colorize the points according to their clusters. You will repeat this process for each criterion. In other words, there will be four colorized plot for a dataset.

2. For each criterion function, give reasons why it is suitable or not suitable for that dataset.

Include the plots and the explanations in your report. Write your code in **task2.py**.

4 Decision Trees (40 pts)

In this section, you will build decision trees with the **ID3** algorithm while trying different feature selection strategies. You will write your code in **task3.py**. The dataset is a CSV file like in the first section and originally taken from here: https://archive.ics.uci.edu/ml/datasets/Nursery. There are 8 features, all of which are categorical. The values of the features and label (class) are given below.

- **parents**: usual, pretentious, great_pret
- **has_nurs**: proper, less_proper, improper, critical, very_crit
- **form**: complete, completed, incomplete, foster
- **children**: 1, 2, 3, more
- **housing**: convenient, less_conv, critical
- **finance**: convenient, inconv
- **social**: nonprob, slightly_prob, problematic
- **health**: recommended, priority, not_recom
- **class**: not_recom, recommend, very_recom, priority, spec_prior

For each of the following selection/pruning strategies, build the decision tree with the training set by using that strategy, state the number of nodes of the resulting tree, and percent accuracy on the test set in your report. Also, comment on how the specific selection/pruning strategy affected the tree in your report. Besides those, for the pruning strategies (item 4 and 5), draw the resulting trees and put them in your report. In the nodes, count of each output class on that current node should be written. If the drawings become unreadable in the report, you may put the images in the submission zip file and reference them in your report.

1. **Information Gain**: Select features according to the Information Gain. For a set S and an attribute A,
$$IG(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i)$$
where E is the entropy.

2. **Gain Ratio**: Select features according to the Gain Ratio. For a set S and attribute A,

$$GR(S,A) = \frac{IG(S,A)}{-\sum_i \frac{|S_i|}{|S|} log_2\left(\frac{|S_i|}{|S|}\right)}$$

3. **Average Gini Index**: Select features according to the Average Gini Index. For a set S and attribute A,

$$\text{Gini Index: } GI(S) = 1 - \sum_i p_i \quad \text{where } p_i \text{ is the proportion of the } i^{th} \text{ class.}$$

$$AGI(S,A) = \sum_i \frac{|S_i|}{|S|} GI(S_i)$$

4. **Average Gini Index with Chi-squared Pre-prunning**: You should stop growing the tree when you cannot reject the null hypothesis which states that there is no association between the two variable(in our case the feature selected by the Average Gini Index and the class variable) using chi-squared test. The selection of the confidence value may change. In this homework, you will use 90% confidence. The critical values can be obtained from a chi-squared table.

5. **Average Gini Index with Reduced Error Post-prunning**: After building the fourth tree, you will also post-prune it. For this part, you will split the training set into training and validation sets to calculate the validation set accuracy to decide whether the pruning enhanced the performance. You cannot use test set for this purpose. Your algorithm can be like this:

```
while a useful replacing can be done:
    [replace a node with a leaf node (subtree replacement)
    whenever it increases the accuracy on the validation set
    and it has no subtree with such property.]
```

5 Specifications

- Falsifying results, changing the composition of training and test data are strictly forbidden, and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reached the results and if they are working correctly.

- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. The violators will be punished according to the department regulations.

- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions on the discussion section of ODTUClass instead of e-mailing.
- You have total of 3 late days for **all** your homeworks. For each day you have submitted late, you will lose 10 points. The homeworks you submit late after your total late days have exceeded 3 will not be graded.

6 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw2.zip** that contains **task1.py**, **task2.py**, **task3.py**, and **report.pdf**.