Middle East Technical University Department of Computer Engineering

CENG 414

Special Topics in Ceng: Introduction to Data Mining

Spring 2020-2021

Programming HW1

Görkem Özer
gorkem@ceng.metu.edu.tr

Due date: May 18th, 2021 Tuesday, 23:59

1 Overview

In this assignment, you are going to use **Weka 3.8** to do some experiments on various datasets. The aim is to make you familiar with certain machine learning algorithms and Weka. Weka is a tool that has collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset through Weka desktop application or they can be called from your own Java code. You will have chance to try both for this assignment.

Keywords: *Weka, Classification, Multi-Layer Perceptron, Decision Tree, SVM, Generalized Sequence Pattern*

2 Tasks

You are expected to complete 4 different main tasks and their subtasks. You will use *"diabetes.arff"*, *"iris.arff"*, *"car.arff"*, *"mnist.arff"*, *"sequence1.arff"*, *"sequence2.arff"* datasets, which will be given to you within homework files.

2.1 Task 1 - Multi-Layer Perceptron (40 Pts.)

For this task, you will use the *"diabetes.arff"* dataset on Weka. Under the *Classify* tab in explorer window, choose *MultilayerPerceptron* classifier. Split the data set into 70% and 30% for training set and test set. You will run the classifier with the default parameters and note them.
Answer the following questions according to the run; **(2 pts. each)**

1. How many hidden layers and hidden nodes created?
2. Did Weka normalize the attributes? What is the effect of normalizing the attributes?
3. What is the benefit of splitting the dataset as training set and test set? Why don't we just train our model with whole data?
4. Which halting strategy did MLP use?

5. What is the detailed accuracy table by class of the run?

Now change the configurations of the MLP by clicking on the name of the classifier. Run the classification task with different training times (100, 300, 500, 1000, 3000, 5000, 10000) while keeping other variables same and plot the **training time-test accuracy plot**. Interpret the accuracy results as; what is the relation between accuracy and training time (epoch count)? What may cause this situation? **(10 pts.)**

It is time to play with learning rate parameter. However, for this part you will use Weka through your Java code. Fill the blanks in the given code. Your code will read *diabetes.arff* file and split it into train and test data, train a MLP from the train data and then evaluate it with the splitted test data. You are expected to complete the functions; *readData, splitData, trainMLP, evaluateMLP*. The signature of the methods and the comments are self explanatory. **Do not change the main method.** Your code will give the detailed accuracy for each predefined hidden layer structure and learning rate. Plot the **learning rate-accuracy plot** for the hidden layer structure (10,15,25). Comment on the trend of the accuracy; what might result in this? In addition, plot the **hidden layer structure-time elapsed plot** for the max epoch of 5000. Comment on the time elapsed and how it is changed by hidden layer structure. **(20 pts.)**
Important: Your code should be compiled and run as follows smoothly without any effort. Make sure you give **absolute paths** while compiling and running your code.

```
_> javac Hw2Q1.java -cp <path of weka>/weka.jar

_> java -cp .:<path of weka>/weka.jar Hw2Q1 <diabetes.arff file path>
```

2.2 Task 2 - Decision Tree (20 Pts.)

Open the explorer in Weka GUI and open the *"iris.arff"*. Go to *Classify* tab and choose *J48 classifier* under trees. Split the data set again into 70% and 30% for training set and test set. Run the classifier without changing default parameters. Report **J48 pruned tree**, **Summary** and **Detailed Accuracy By Class**. In addition, put the visualization of the tree under task2/Docs folder. Answer the following questions;

1. What is the meaning of information gain?

2. What is the entropy of each non-leaf node in the result tree? Show your calculations step by step, instead of writing final entropy values only.

3. Trace the tree for predicting the correspondent classes for each given sample below

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
7.2	3.2	5.2	1.5	
3.1	2.7	9.0	0.5	
4.2	5.8	4.7	0.7	
1.4	3.4	3.2	1.9	

2.3 Task 3 - Data Visualization (20 Pts.)

You will visualize and analyse *car.arff* through Weka GUI. Open the dataset and switch to the *Visualize* tab. By changing some properties under the window, analyse the dataset. Give brief answers and explanatories for each question;

1. What does linearly-separable mean?
2. By only using *price* and *maintenance-cost* attributes, is it linearly separable?
3. Is this dataset linearly-separable with all of its attributes? How did you decide?
4. Which classifier would you use to classify this dataset? Explain briefly.
    - Logistic regression
    - MLP
    - Naive Bayes
    - J48 Tree

2.4 Task 4 - Support Vector Machine (SVM) (20 Pts.)

In this task, you will use SVM, which is a very popular classifier. First, open the *mnist.arff* dataset in Weka GUI. Under the *Classify* tab, choose *SMO(Support Vector Machine implementation).* If it is disabled, it is probably because of the class label. SVM works on discrete data labels. Therefore, you need to convert numeric class label to nominal from *Preprocess* tab. After converting the class label to nominal, choose SMO under the *classfiers.functions.* Run the classifier with default paramaters and report **Summary** and **Detailed Accuracy By Class**. Explain what is the C parameter of SVM and how it effects on this dataset by changing it with experimental results.

3 Submission

1. For each task create a directory named **task-1**, **task-2**, **task-3**, **task-4**. All of your solutions, codes, comments, plots about a task should be inside the correspondent directory. If your directory structure is messy, you will get **penalty**.
2. Under each task directory, you should have **Source** and **Docs** directories. Any source code should be under **Source** and the others in **Docs**.
3. Your Java code should be tested on inek machines before submitting.
4. Do not put your binary files, IDE related files etc. Only Java files will be submitted.
5. Zip all task directories and name it as <ID> _ <FullNameSurname> and submit it through COW. **Replace with your ID number and your name.** For example:

   `e1234567_GorkemOzer.zip`

6. Late submissions will **NOT** be accepted.