Middle East Technical University   Department of Computer Engineering

CENG 414
Introduction to Data Mining
Fall 2017-2018

Assignment 2

Yusuf Mücahit Çetinkaya
yusufc@ceng.metu.edu.tr

Due date: November 28, 2017, 23:59

1 Overview

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

2 Tasks

You are expected to complete 5 different tasks, which may have subtasks each. You can download *"diabetes.arff"*, *"iris.arff"*, *"car.arff"*, *"mnist.arff"*, *"sequence1.arff"* and *"sequence2.arff"* datasets from course webpage on COW.

2.1 Multilayer 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 and test. 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. Which halting strategy did MLP use?
4. 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 the training times (100, 300, 500, 1000, 3000, 5000, 10000) by keeping other variables same and plot the epoch count-test accuracy plot. Interpret the accuracy results as; what is the relation between accuracy and 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 **.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.**(22 pts.)**
Important: Your code should be compiled and run as follows smoothly without any effort;

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

2.2 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 and test. 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 information gain?
2. What is the entropy of each non-leaf node in the result tree?
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 Data Visualization (10 Pts.)

You will visualize and analyse *car-dataset.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 *maint-cost* attributes, is it linearly seperable?
3. Is this dataset linearly-seperable with all of its attributes? How did you decide?

4. Which classifier would you use to classify this dataset?
    - Logistic regression
    - MLP
    - Naive Bayes
    - J48 Tree

2.4 Generalized Sequence Pattern (15 Pts.)

In this task, you will detect sequence patterns by applying *Generalized Sequence Patterrn* algorithm to the dataset sequence1.arff and sequence1.arff through your own Java code. This algorithm is not located in weka.jar in default. Therefore, you should download the external jar manually. You can find the algorithm **here**.
Complete the given code and run GSP algorithms on both data sets with 0.3, 0.5 and 0.7 support thresholds. Record the frequent patterns for each support and execution time for each run. Plot execution time (in ms)/support threshold in the form of bar chart for both datasets. Comment on the frequent patterns extracted and the duration of execution for each run you made.
Important: Your code should be compiled and run as follows smoothly without any effort;

```
>> javac Hw2Q4.java -cp <path of weka>/weka.jar:<path of gsp>/generalizedSequentialPatterns.jar
>> java -cp .:<path of weka>/weka.jar:<path of gsp>/generalizedSequentialPatterns.jar Hw2Q4
<sequence1.arff file path> <sequence2.arff file path>
```

2.5 Support Vector Machine (15 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 paramateres 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-5**. 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 codes should be tested on inek machines before submit.
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. For example:
   e1234567_YusufMucahitCetinkaya.zip

Note: Any extra effort will be rewarded. **Late submissions will not be accepted.**