CS 565 Project 1: Simple Data Preprocessing Using Weka

(due on the class on 10/05/16)

In this project, you will start to learn how to use the software Weka for data processing. Weka is an open software that can be downloaded from the website http://www.cs.waikato.ac.nz/ml/weka/. Weka has been installed in the computers in the CS Lab. Weka is implemented in Java. Weka includes a GUI (graphical interface) as a platform for data mining. But more essentially, Weka is a Java library consisting of a set of Classes for various data mining tasks, such as data preprocessing, classification, association, and clustering. You can easily integrate Weka into a specific Java application. But you will need to set the classpath correctly in order to use the Weka library during the compilation and execution. The major file in Weka is weka.jar. Assume that you are using a Windows system and weka.jar is stored in the directory

```
C:\program files\weka-3-6\
```

You may use the following statement to set a classpath

```
> set CLASSPATH=.;C:\program files\weka-3-6\weka.jar
```

If you use eclipse for Java programming, you may follow the following steps:

1. Create a new project in Eclipse. Under "Package Explorer" $->$ Right Click project $->$ Properties.
2. Select "Java Build Path" $->$ "Libraries"
3. "Add External Jars" and select the weka.jar from the above directory.

1 Part I: Statistics of a dataset

In Weka, a dataset is recorded in the format called **ARFF**. The following is an example of a dataset (weather.numeric.arff in the data sub-directory of weka software):

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
```

```
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Write a Java program to output the following statistics of dataset:

1. For a nominal attribute
   (a) output the number of possible attribute values
   (b) the number of instances for each attribute value
2. For a numeric attribute
   (a) output the maximum and minimum value
   (b) split the range from minimum to maximum into two equal-length intervals (say lower and upper), and output the number of instances lower and upper intervals, respectively.

You will need to read the documentation of the package called **weka.core**, and typically the class **Attribute** to complete the this part.

2 Part II: Euclidean distance vs. Mahalanobis distance

In the class, we have learned that distance measurement is an important topic in data mining. Given two data points x and y whose attributes are all numeric, the Euclidean distance $\|x - y\|$ is a popular measurement. To be consistent with the textbook, we assume a vector x by default is a row vector.

However, Euclidean distance is not effective if

- the ranges (scales) of attributes are different, or
- there exist correlations among attributes.

Mahalanobis distance has been introduced to address the above problems. Given a dataset whose attributes are all numeric, we first construct the the covariance matrix $\Sigma = (s_{i,j})_{n\times n}$, where an element $s_{i,j}$ is the covariance of i-th and j-th attributes, and n is the number of attributes. The Mahalanobis distance of x and y is then defined as

$$Mahalanobis(x, y) = (x - y)\Sigma^{-}(x - y)^T,$$

where Σ^{-} is the inverse of the matrix Σ, and $(x - y)^T$ is the transpose of the row vector $x - y$.

In the part, you will first implement the above two distance measurements, and then give an experimental evaluation of these measurements. Usually, evaluation is performed in a specific data mining tasks, such as classification, clustering. Since we currently have not yet got into these topics, we will simply check the consistency between two measurements based on random datasets. Specifically, you will need to

1. generate a random dataset of two-attribute instances in Gaussian distribution
2. build the covariance matrix Σ
3. set counter = 0
4. for each instance,
   (a) Compute the nearest neighbor using Euclidean distance
   (b) Compute the nearest neighbor using Mahalanobis distance
   (c) if the nearest neighbors are same, count++
5. output the consistency ratio (count / number-of-instances)

You will need to read the documentation of the packages **weka.core** and **weka.core.matrix**, and typically the class **Matrix** to complete this part. To generate data in Gaussian distribution, you may refer to the standard Java class Random, and use the Java method nextGaussian().

This project is a team project, and you can form a group of at most two.

On submission: Submit the source code to Blackboard. The source code should be well-documented, i.e., having good readability. The readme file should summarize: i) tasks that you completed, and ii) tasks that you did not complete.