Use WEKA in your Java code

TA: Binh Tran

The most common components you might want to use, are

- Instance - your data
- Filer – for pre-processing data
- Classifier/Clusterer - is built on the processed data
- Evaluating – how good is the classifier/clusterer?
- Attribute selection – removing irrelevant attributes from your data

The following section is an example to show **how to call Weka decision tree J48** from your Java code, *using default parameters and using with options*

The necessary classes can be found in this package:

weka.classifiers

We can train the J48 tree algorithm, a Weka classifier, on a given dataset *data*. The training is done via the **buildClassifier (Instances)** method.

The java code needs look something like this,

```
import weka.classifiers.trees.J48;
 ...

J48 tree = new J48();          // new instance of tree
tree.buildClassifier(data);    // build classifier
```

Notes on Database

Reading from an ARFF file is straight forward:

```
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
...
BufferedReader reader = new BufferedReader(
                            new FileReader("/some/where/data.arff"));
Instances data = new Instances(reader);
reader.close();
// setting class attribute
data.setClassIndex(data.numAttributes() - 1);
```

The class index indicate the target attribute used for classification. By default, in an ARFF File, it's the last attribute, that's why it's set to numAttributes-1.
You **must** set it if your instances are used as a parameter of a weka function (ex: `weka.classifiers.Classifier.buildClassifier(data)`)

Notes option handling

Weka schemes that implement the `weka.core.OptionHandler` interface, like classifiers, clusterers, filters, etc. offer the following methods for setting and retrieving options:

- `void setOptions(String[] options)`
- `String[] getOptions()`

There are several ways of setting the options:

- Manually creating a String array:

```
String[] options = new String[2];
options[0] = "-R";
options[1] = "1";
```

- Using a single commandline string and using the `splitOptions` method of the `weka.core.Utils` class to turn it into an array:

```
String[] options = weka.core.Utils.splitOptions("-R 1");
```

For examples, calling J48 with minimum number objects =5, confidence Factor = 1.0,the java code is as follows

```
import weka.classifiers.trees.J48;
...
String[] options = new String[1];
options[0] = "-C 1.0 -M 5";     // confidenceFactor = 1.0, minNumObject = 5
J48 tree = new J48();          // new instance of tree
tree.setOptions(options);      // set the options
tree.buildClassifier(data);    // build classifier
```

References

1. http://weka.wikispaces.com