1 Wekalink: using an AMIDST classifier in Weka

One of the greatests points of AMIDST is the integration with other tools for data analysis. This is the case of Weka whose integration fuctionality is provided by the module *wekalink*. We will be able to create a wrapper for evaluating an AMDIST classifier with Weka.

1.1 Prepare your project

The first thing we have to do is to load the required AMIDST dependencies in a Maven project. In this case, we will have to load the modules **wekalink** and **latent-variable-models**. For that, add the following code to the file pom.xml of your project.

```
<dependencies>

  <dependency>
    <groupId>eu.amidst</groupId>
    <artifactId>wekalink</artifactId>
    <version>0.4.3-alpha</version>
    <scope>compile</scope>
 </dependency>

  <dependency>
    <groupId>eu.amidst</groupId>
    <artifactId>latent-variable-models</artifactId>
    <version>0.4.3-alpha</version>
    <scope>compile</scope>
  </dependency>

<!-- ... -->
</dependencies>
```

Further details for creating a project using AMIDST fuctionality is given in the Getting Started section.

1.2 Create the wrapper class

A custom classifier that could be handled by weka should inherit from class *weka.classifiers.AbstractClassifier* and implement interface *weka.core.Randomizable*. Thus we should override at least the following methods:

- *void buildClassifier(Instances data)*: builds the classifier from scratch with the given dataset.
- *double[] distributionForInstance(Instance instance)*: returns a vector containing the probability for each label or state of the class.

Here below we show a minimal example where the Naive Bayes classifier provided by AMIDST is used.

```
import eu.amidst.core.datastream.Attributes;
import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataOnMemoryListContainer;
import eu.amidst.core.datastream.filereaders.DataInstanceFromDataRow;
import eu.amidst.latentvariablemodels.staticmodels.classifiers.NaiveBayesClassifier;
import eu.amidst.wekalink.converterFromWekaToAmidst.Converter;
import eu.amidst.wekalink.converterFromWekaToAmidst.DataRowWeka;
import weka.classifiers.AbstractClassifier;
import weka.core.Instance;
import weka.core.Instances;


public class AmidstNaiveBayes extends AbstractClassifier {

        private NaiveBayesClassifier model = null;
        private Attributes attributes;

        @Override
        public void buildClassifier(Instances data) throws Exception {

                attributes = Converter.convertAttributes(data.enumerateAttributes(),
                                                        data.classAttribute());
                DataOnMemoryListContainer<DataInstance> dataAmidst =
                                        new DataOnMemoryListContainer(attributes);

                data.stream()
                        .forEach(instance -> dataAmidst.add(
                                new DataInstanceFromDataRow(
                                        new DataRowWeka(instance, attributes)))
                );

                model = new NaiveBayesClassifier(attributes);
                model.updateModel(dataAmidst);
        }


        @Override
        public double[] distributionForInstance(Instance instance) throws Exception {
                DataInstance amidstInstance =
                        new DataInstanceFromDataRow(new DataRowWeka(instance,
                                                        this.attributes));
                return model.predict(amidstInstance).getParameters();

        }

}
```

Note that previous code does not implement neither the learning nor the classification processes, it simply calls to the corresponding methods

eu.amidst.latentvariablemodels.NaiveBayesClassifier performing such tasks.

1.3 Testing the AMIDST classifier in Weka

Now we can evaluate an AMIDST classifier using only calls to functions from Weka. Here we show an example where we load a dataset in format .arff, we learn a naive Bayes classifier and we show the confusion matrix:

```
//Load the dataset
BufferedReader reader =
        new BufferedReader(new FileReader("exampleDS_d5_c0.arff"));
Instances data = new Instances(reader);
data.setClassIndex(6);

//Learn and evaluate the classifier
Evaluation eval = new Evaluation(data);
Debug.Random rand = new Debug.Random(1);
int folds = 10;
Classifier cls = new AmidstNaiveBayes();
eval.crossValidateModel(cls, data, folds, rand);

//Print the confusion matrix
System.out.println(eval.toMatrixString());
```