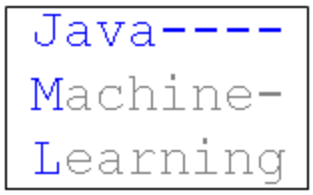

Published on *Java Machine Learning Library (Java-ML)* (http://java-ml.sourceforge.net)

By *Thomas Abeel*
Created *11/27/2008 - 13:41*

Getting started

This collection of articles will guide you through the basics of the Java Machine Learning Library.

In these articles you will learn how to install the library, the terminology used in the library and the documentation, how to load data, how to construct classifiers, filters, and other algorithms and how you can apply them to your data.

Finding documentation

Where do I find the documentation and manual for this library?

Website
There are several places to find documentation for our library. First of all you have our website (http://java-ml.sourceforge.net [1]). At this website you can find:

- Getting started chapter [2]: this will introduce you to the basics and help you install the library
- Collections of articles on classification [3], feature selection [4] and clustering [5].
- Developer documentation [6].
- API reference documentation [7].

Manual
Although you can find the most recent documentation on our website, you may want to have more static documentation that goes along with your preferred release. All articles are exported to pdf with each release, they are available from the download repository [8].

Installing the library

A step-by-step guide to install Java-ML.

1. Download the latest version of the library from Sourceforge:

http://downloads.sourceforge.net/java-ml/ [8]

2. Unpack the downloaded file in a directory of your choice, preferable in the one you will be developing your machine learning applications.
3. The library consists of a main jar-file (javaml-<version>.jar) and a number of support libraries that reside in the *lib/* directory. For ease of use, you have to add all of them to your classpath, both the main Java-ML jar and the supporting libraries. If you are using an IDE like Eclipse or Netbeans, you have to add them to the build-path.
4. Once you have include Java-ML and the supporting libraries in your classpath, you can start using the library

Basic terminology

This article explains the various terms that are used throughout the rest of the tutorials and are also used in the source-code documentation.

We assume that you are somewhat familiar with the concept of machine learning [9] or data mining [10].

We also assume you have at the very least a basic knowledge of Java, the language in which the library is written.

In short every data sample is stored in an Instance, which are grouped together in a Dataset. Each Instance can have a number of attributes that have real values. The terms features and attributes are synonymous in this context.

Each Instance can have a class label. Algorithms are functions that work on Datasets and Instances. Classification algorithms for example can be trained on a Dataset and can later be applied to classify Instances.

More in detail:

Instance
: The representation of real world sample. An instance can have any number of attributes that define it and can have at most one class label.

Dataset
: A dataset is a collection of instances that belong together.

Algorithm
: Any method or technique that can work with Datasets and/or Instances. This can be algorithms for classification, clustering, regression, filtering, etc.

Creating an Instance

This tutorial shows the very first step in using Java-ML. It will show you how to create Instances which are the default way to represent a sample of real world data.

The main way to represent data is the DenseInstance which requires a value for each attribute of an Instance. In most scenarios this representation of the data will suffice. In case your data

is sparse, you can also put your data in a SparseInstance which requires less memory in case of sparse data (less than 10% attributes set).

How to create a DenseInstance
[Documented source code] [11]

The block of code below will create an Instance with ten attributes. The values of these attributes are one through ten. The class label of this Instance is not set.

```
.     /* values of the attributes. */
.     double[] values = new double[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
.     /*
.      * The simplest incarnation of the DenseInstance constructor will only
.      * take a double array as argument an will create an instance with given
.      * values as attributes and no class value set. For unsupervised machine
.      * learning techniques this is probably the most convenient constructor.
.      */
.     Instance instance = new DenseInstance(values);
```

To create an Instance with ten attributes, with values one through ten and with the class label "positive" we can use the following code. Note that the class label can be any object.

```
.     /* values of the attributes. */
.     double[] values = new double[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
.     /* Create and an Instance with the above values and its class label set to "
.     Instance instance = new DenseInstance(values,"positive");
```

How to create a SparseInstance
[Documented source code] [12]
To create a SparseInstance you just have to specify the number of attributes that the Instance has and then you set the particular attributes values. The example below creates an Instance with 10 attributes. Next, we set the value of attributes with index one, three and seven to particular values.

```
.     /*
.      * Here we will create an instance with 10 attributes, but will only set
.      * the attributes with index 1,3 and 7 with a value.
.      */
.     /* Create instance with 10 attributes */
.     Instance instance = new SparseInstance(10);
.     /* Set the values for particular attributes */
.     instance.put(1, 1.0);
.     instance.put(3, 2.0);
.     instance.put(7, 4.0);
```

Creating a Dataset

This tutorial show how to create a Dataset. Simply put, a Dataset is a collection of Instances.

This tutorial assumes you know how to create an Instance, either DenseInstance or SparseInstance will work as both are implementations of the Instance interface.

For the purpose of this tutorial we will use a method of InstanceTools to create Instances with random values for its attributes. In reality you will either create the Instances yourself or you will load them from a file (see next page in this tutorial trail). The method *InstanceTools.randomInstance(25);* will create a DenseInstance with 25 attributes that all have values between zero and one.

Now on to creating a Dataset. Dataset is an interface which defines a number of operations on a data set. The default implementation of Dataset is DefaultDataset. At the moment no other implementations of Dataset are available. In the following example we will create a DefaultDataset and populate it with 10 Instances that have been randomly generated as described in the previous paragraph.

```
. Dataset data = new DefaultDataset();
. for (int i = 0; i < 10; i++) {
.     Instance tmpInstance = InstanceTools.randomInstance(25);
.     data.add(tmpInstance);
. }
```

[Documented source code] [13]

Load data from file

This tutorial shows how you can read data from a number of file types. At the moment the library supports any type of field formatted file, i.e. one sample on a line and the attributes separated by a symbol (commonly , (comma) ;(semi-colon) or \t (tab)). Common formats include CSV and TSV and this type of file is further denoted by 'CSV-like file'. Java-ML also supports ARFF formatted files, with the limitation that only the class label can be non-numeric.

Loading data from CSV-like files
[Documented source code] [14]

```
. Dataset data = FileHandler.loadDataset(new File [15]("iris.data"), 4, ",");
```

The first parameter of loadDataset is the file to load the data from. The second parameter is the index of the class label (zero-based) and the final parameter is the separator used to split the attributes in the file.

Data for the above example.
Iris data set [16]

Loading sparse data from CSV-like files
[Documented source code] [14]

```
. Dataset data =  FileHandler.loadSparseDataset(new File [15]("sparse.tsv"), 0, " ",
```

The first parameter of loadSparseDataset is the file to load the data from. The second parameter is the index of the class label (zero-based), the third parameter is the attribute separator and the final parameter is the separator used to split the index and value of the attribute.

Sample data file for sparse data
Sample sparse data [17]

Loading data from an ARFF formatted file
Loading from an ARFF formatted file is much like loading from a CSV or TSV file.
[Documented source code] [18]

```
. Dataset data = ARFFHandler.loadARFF(new File [15]("iris.arff"), 4);
```

The loadARFF method of the ARFFHandler only has two arguments, the first one to indicate the file that should be loaded and the second one to indicate the index of the class label. The ARFFHandler also has a method with only a File as argument to load files that do not contain a class label.

There are some caveats with the ARFF loader: (i) all the header information is ignored by the loader, and (ii) Java-ML only supports numeric attributes, data sets that have attributes that are not numeric will crash the loader.

Sample data file for ARFF loader
Sample Iris ARFF data [19]

Note that the FileHandler works with flat files or GZIP compressed files, while the ARFFHandler only works with flat files.

Store data to file

Java-ML can store Datasets back to file using the FileHandler class. At the moment it is not possible to store data in the ARFF format.

In the previous article in the tutorial trail we have shown how to load data from a file, we will use this mechanism again to load our data and then store it to a different file.

Storing data to a file
[Documented source code] [20]

```
. /* Load the iris data set from file */
. Dataset data = FileHandler.loadDataset(new File [15]("iris.data"), 4, ",");
. System [21].out.println(data);
. /* Store the data back to another file */
. FileHandler.exportDataset(data,new File [15]("output.txt"));
```

The above example will load the iris data set and will store it back to another file (output.txt). Each instance is written to the file separately with the class label on position 0. The fields are delimited with a tab character.

The data that was outputted can be read again with the following code:

```
. Dataset data = FileHandler.loadDataset(new File [15]("output.txt"), 0,"\t");
```

Caveats: Note that data sets with mixed sparse and dense instances may result in files that cannot be loaded with the methods in FileHandler. Exporting data sets where some Instances have their class label set and some don't, will result in unwanted behaviour when loading the file with the methods in FileHandler.

Source URL (retrieved on *06/05/2009 - 10:44*): http://java-ml.sourceforge.net/content/getting-started

Links:
[1] http://java-ml.sourceforge.net
[2] http://java-ml.sourceforge.net/content/getting-started
[3] http://java-ml.sourceforge.net/content/classification
[4] http://java-ml.sourceforge.net/content/feature-selection
[5] http://java-ml.sourceforge.net/content/clustering
[6] http://java-ml.sourceforge.net/content/developer-documentation
[7] http://java-ml.sourceforge.net/api
[8] http://downloads.sourceforge.net/java-ml/
[9] http://en.wikipedia.org/wiki/Machine_learning
[10] http://en.wikipedia.org/wiki/Data_mining
[11] http://java-ml.sourceforge.net/src/tutorials/core/TutorialDenseInstance.java
[12] http://java-ml.sourceforge.net/src/tutorials/core/TutorialSparseInstance.java
[13] http://java-ml.sourceforge.net/src/tutorials/core/TutorialDataset.java
[14] http://java-ml.sourceforge.net/src/tutorials/tools/TutorialDataLoader.java
[15] http://www.google.com/search?hl=en&q=allinurl:file java.sun.com&btnI=I'm Feeling Lucky
[16] http://java-ml.svn.sourceforge.net/viewvc/java-ml/trunk/devtools/data/iris.data
[17] http://java-ml.svn.sourceforge.net/viewvc/java-ml/trunk/devtools/data/sparse.tsv
[18] http://java-ml.sourceforge.net/src/tutorials/tools/TutorialARFFLoader.java
[19] http://java-ml.svn.sourceforge.net/viewvc/java-ml/trunk/devtools/data/iris.arff
[20] http://java-ml.sourceforge.net/src/tutorials/tools/TutorialStoreData.java
[21] http://www.google.com/search?hl=en&q=allinurl:system java.sun.com&btnI=I'm Feeling Lucky