

Machine Learning with Weka: Using ML in Java

PU Tools, Ressourcen, Infrastruktur

Nils Reiter,
`nils.reiter@uni-koeln.de`

December 17, 2020
(Winter term 2020/21)

Exercise 6

<https://github.com/idh-cologne-tools-ressourcen-infra/exercise-06>

Section 2

Weka as Java API

Java

- ▶ Object-oriented programming (OOP)
- ▶ Platform-independent via virtual machine
- ▶ Complicated licensing situation
 - ▶ Sun bought by Oracle
- ▶ OpenJDK: <https://adoptopenjdk.net>

Java

- ▶ Object-oriented programming (OOP)
- ▶ Platform-independent via virtual machine
- ▶ Complicated licensing situation
 - ▶ Sun bought by Oracle
- ▶ OpenJDK: <https://adoptopenjdk.net>
- ▶ Classes are organised in packages (\simeq folders)
- ▶ Compiled code can be packaged in jar files
- ▶ Jar files can be added to the classpath, and then used
- ▶ Basics: <https://github.com/DH-Cologne/java-wegweiser>
(Börge Kiss & collaborators)

java.util...

Introduction

- ▶ Weka can be used as a Java API
- ▶ Not a lot of tutorial documentation
- ▶ Javadoc: <https://javadoc.io/doc/nz.ac.waikato.cms.weka/weka-stable/latest/index.html>

Introduction

- ▶ Weka can be used as a Java API
- ▶ Not a lot of tutorial documentation
- ▶ Javadoc: <https://javadoc.io/doc/nz.ac.waikato.cms.weka/weka-stable/latest/index.html>
- ▶ Download
 - ▶ Direct: `weka.jar` from https://waikato.github.io/weka-wiki/downloading_weka/
 - ▶ Maven: `nz.ac.waikato.cms.weka / weka-stable / 3.8.4`

Beware: Coding Machine Learning

- ▶ Using machine learning in code relies on libraries
- ▶ Libraries (here: Weka) do all the (probabilistic|numerical|statistical) ›heavy lifting‹

Beware: Coding Machine Learning

- ▶ Using machine learning in code relies on libraries
- ▶ Libraries (here: Weka) do all the (probabilistic|numerical|statistical) ›heavy lifting‹
- ▶ What remains?
 - ▶ Input
 - ▶ Derive suitable input representation
 - ▶ Prepare data to be in the right format
 - ▶ Output
 - ▶ Do something with the result or the model

Beware: Coding Machine Learning

- ▶ Using machine learning in code relies on libraries
- ▶ Libraries (here: Weka) do all the (probabilistic|numerical|statistical) ›heavy lifting‹
- ▶ What remains?
 - ▶ Input
 - ▶ Derive suitable input representation
 - ▶ Prepare data to be in the right format
 - ▶ Output
 - ▶ Do something with the result or the model

⇒ Code distribution

- ▶ Most of the code will not be about machine learning
- ▶ But about input/output handling
- ▶ This also consumes most of the development time

API Overview

Packages

- ▶ `weka.core`: Data IO, some preprocessing, conversion
- ▶ `weka.gui`: Everything related to the GUI
- ▶ `weka.filters`: Filters. Further subdivision into feature vs. instance and super- vs. unsupervised
- ▶ `weka.classifiers`: Classifiers. Further subdivision into types
- ▶ `weka.clusterers`: Clustering methods. Not so many.

Most Important Classes

- ▶ `weka.core.Instances`: Container class for a data set, supports weighting and ordering
- ▶ `weka.core.Instance`: Interface for a single instance. Classes are: `DenseInstance`, `SparseInstance`, `BinarySparseInstance`
- ▶ `weka.core.Attribute`: Represents a feature (not a feature value!)
 - ▶ Attributes have a name and a data type (numeric, nominal, string, date)
- ▶ `weka.classifiers.Classifier`: Interface that all classifiers implement
- ▶ `weka.filters.Filter`: Interface that all filters implement

Instance Representation

- ▶ Internally, all (non-string) attribute values are represented as double values – even nominal values!
 - ▶ Mapping nominal values is done by the class
 - ▶ Methods to set attribute values come in two variants:
void setValue(Attribute att, String value) and
void setValue(Attribute att, double value)

Instance Representation

- ▶ Internally, all (non-string) attribute values are represented as double values – even nominal values!
 - ▶ Mapping nominal values is done by the class
 - ▶ Methods to set attribute values come in two variants:
`void setValue(Attribute att, String value)` and
`void setValue(Attribute att, double value)`

DenseInstance vs. SparseInstance

- ▶ `DenseInstance`: The naive implementation, representing each feature value as a double
- ▶ `SparseInstance`: Stores only non-zero values
 - ▶ Nominal values: First value not stored (because it has index zero)
- ▶ `SparseInstance` useful for large data sets with many default values

<https://javadoc.io/static/nz.ac.waikato.cms.weka/weka-stable/3.8.4/weka/core/Instances.html>

weka.classifiers.Classifier

Methods

- ▶ Train a classifier

```
public void buildClassifier(Instances data) throws Exception
```

- ▶ Classify a single instance

```
public double classifyInstance(Instance instance) throws Exception
```

- ▶ return value represented as double!

- ▶ Classify a single instance

```
public double[] distributionForInstance(Instance instance) throws Exception
```

- ▶ May return a distribution of class probabilities

- ▶ Meta data about the classifier

```
public Capabilities getCapabilities()
```

<https://javadoc.io/static/nz.ac.waikato.cms.weka/weka-stable/3.8.4/weka/classifiers/Classifier.html>

weka.filters.Filter

Methods

- ▶ Filter a single instance

```
public boolean input(Instance instance) throws Exception
```

- ▶ Retrieve output

```
public Instance output()
```

- ▶ Filter an entire data set

```
public static Instances useFilter(Instances data, Filter filter) throws Exception
```

- ▶ ...

<https://javadoc.io/static/nz.ac.waikato.cms.weka/weka-stable/3.8.4/weka/filters/Filter.html>


```
1 class Main {
2     public static void main(String[] args) throws IOException, Exception
3
4         // load data
5         ArffLoader loader = new ArffLoader();
6         loader.setFile(new File("data.arff"));
7         Instances instances = loader.getDataSet();
8         instances.setClassIndex(instances.numAttributes() - 1);
9
10        // train classifier
11        Naive Bayes classifier = new OneR(); filter
12        classifier.buildClassifier(instances); model to file
13
14        // run cross validation
15        Evaluation eval = new Evaluation(instances);
16        eval.crossValidateModel(classifier, instances, 10, new Random());
17    }
18 }
```

References I