Programmatic Use (/Programmatic+Use)

Table of Contents

Introduction
Step 1: Express the problem with features
Step 2: Train a Classifier
Step 3: Test the classifier
Step 4: use the classifier
Conclusion and More Information
Links

Introduction

This tutorial shows how to use Weka (build **feature vector**, **train** a classifier, **test** a classifier, **use** a classifier) directly from Java code. It is not intended to replace the Explorer/Experimenter GUI that offer the visualization and engineering tools required to set up and debug machine learning experiments. Weka's automation is useful to embed a classifier in a larger program and to create a training/testing loop that can be seen as a regression test for machine learning capabilities.

Step 1: Express the problem with features

This step corresponds to the engineering task needed to

write an *.arff* file.
Let's put all our features in a weka.core.FastVector. Each feature is contained in a weka.core.Attribute object.

Here, we have two numeric features, one nominal feature (blue, gray, black) and a nominal class (positive, negative).

```
// Declare two num
Attribute Attribute1
Attribute Attribute2

// Declare a nomina
FastVector fvNomir
fvNominalVal.addE
fvNominalVal.addE
fvNominalVal.addE
Attribute Attribute3

// Declare the class
FastVector fvClassV
fvClassVal.addElem
fvClassVal.addElem
Attribute ClassAttri

// Declare the featu
FastVector fvWekaA
fvWekaAttributes.a
fvWekaAttributes.a
fvWekaAttributes.a
fvWekaAttributes.a
```

Step 2: Train a Classifier

Training requires 1) having a training set of instances and 2) choosing a classifier.

Let's first create an empty training set (weka.core.Instances). We named the relation "Rel". The attribute prototype is declared using the vector from step 1. We give an initial set

capacity of 10. We also declare that the class attribute is the fourth one in the vector (see step 1)

```
//
In
//
is
```

Now, let's fill the training set with one instance (weka.core.Instance):

```
//
In
iE
iE
iE
iE
```

```
// is
```

Finally, Choose a classifier (weka.classifiers.Classifier) and create the model. Let's, for example, create a naive Bayes classifier (weka.classifiers.bayes.NaiveBayes)

```
// C
c
```

Step 3: Test the classifier

Now that we create and

trained a classifier, let's test it. To do so, we need an evaluation module (weka.classifiers.Evaluation) to which we feed a testing set (see section 2, since the testing set is built like the training set).

```
//
E
e
```

The evaluation

module can output a bunch of statistics:

```
//
S
S

//
d
```

Step 4: use the classifier

For real world applications, the actual use of the classifier is the ultimate goal. Here's the simplest way

to achieve that. Let's say we've built an instance (named *iUse*) as explained in step 2:

```
//
//
iU

//
//
//
d
```

Conclusion and More Information

This tutorial shows the basic way to train, test

and use a classifier programmatically in Weka. The code shown was not compiled nor tested since it requires being part of a real classification problem. For complete and compilable examples, please check Balie , an open source NLP software that uses Weka for language identification

and sentence boundary recognition tasks.

Links

- Weka API (book version /developer version)
- Balie