Java----
Machine-
Learning

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

By *Thomas Abeel*
Created *12/01/2008 - 12:50*

Classification

This chapter provides an introduction to tasks related to classification.

This set of articles assumes that you are familiar with the basics of Java-ML as detailed in the Getting started trail.

Classification basics

This tutorial explains the basics of setting up a classifier, training the algorithm and evaluating its performance. First we need to initialize a classifier, next we can train it with some data, and finally we can use it to classify new instances.

Creating a classifier
The following sample loads data from the iris data set, next we construct a K-nearest neighbor classifier and we train it with the data.

```
. /* Load a data set */
. Dataset data = FileHandler.loadDataset(new File [1]("devtools/data/iris.data"), 4
.  /* Contruct a KNN classifier that uses 5 neighbors to make a
.   *decision. */
. Classifier knn = new KNearestNeighbors(5);
. knn.buildClassifier(data);
```

[Documented source code] [2]

Note that the build method of a classifier may modify the Dataset that is provided as parameter.

Evaluating the performance of a classifier
Now that we have constructed and trained a classifier, we can use it to classify new Instances. In this example we will reload the iris data set and use the trained classifier to predict the class label for each Instance.

```
.  Dataset dataForClassification = FileHandler.loadDataset(new File [1]("devtools/d
. /* Counters for correct and wrong predictions. */
. int correct = 0, wrong = 0;
. /* Classify all instances and check with the correct class values */
. for (Instance inst : dataForClassification) {
.     Object [3] predictedClassValue = knn.classify(inst);
.     Object [3] realClassValue = inst.classValue();
.     if (predictedClassValue.equals(realClassValue))
.         correct++;
.     else
.         wrong++;
. }
```

This example will go over all instances in the iris data set and try to predict its class by majority voting on its 5 neighbors. In this example this will result in 145 correct predictions and 5 wrong ones.

Note that this is not the proper way to do validation of a classifer. For the proper technique, look at cross validation [4].

Evaluate classifier on a dataset

This tutorial shows you how you can test the performance of a classifier on a data set. This tutorial will introduce two classes. EvaluateDataset, which allows you to test a classifier on a data set and it will also introduce PerformanceMeasure. This class is used to store information regarding the performance of a classifier.

Evaluate a classifier on a dataset

```
. Dataset data = FileHandler.loadDataset(new File [1]("devtools/data/iris.data"), 4
. Classifier knn = new KNearestNeighbors(5);
. knn.buildClassifier(data);
. Dataset dataForClassification = FileHandler.loadDataset(new File [1]("devtools/da
.
. Map<Object, PerformanceMeasure> pm = EvaluateDataset.testDataset(knn, dataForCla
. for(Object [3] o:pm.keySet())
.     System [5].out.println(o+": "+pm.get(o).getAccuracy());
```

[Documented source code] [6]
This sample loads the iris data set, constructs a 5-nearest neighbor classifier and loads the iris data again.

The testDataset method will use the trained classifier to predict the labels for all instances in the supplied data set. The performance of the classifier is returned as a map that contains for each class a performance measure. A PerformanceMeasure is a wrapper around the values for the true positives, true negatives, false positives and false negatives. This class also provides a number of convenience method to calculate a number of aggregate measures like

accuracy, f-score, recall, precision, sensitivity, specificity, etc.

Classification cross validation

In this tutorial we discuss how you can perform cross-validation with Java-ML.

In this tutorial we assume that you know how to load data from a file [7], how to create a classifier [8] and how to work with the PerformanceMeasure.

Cross validation in Java-ML can be done using the CrossValidation class. The code below shows how to use this class.

[Documented source code] [9]

```
. /* Load data */
. Dataset data = FileHandler.loadDataset(new File [1]("iris.data"), 4, ",");
. /* Construct KNN classifier */
. Classifier knn = new KNearestNeighbors(5);
. /* Construct new cross validation instance with the KNN classifier */
. CrossValidation cv = new CrossValidation(knn);
. /* Perform cross-validation on the data set */
. Map<Object, PerformanceMeasure> p = cv.crossValidation(data);
```

This example first loads the iris data set and then constructs a K-nearest neighbors classifier that uses 5 neighbors to classify instances.
In the next step we create a cross-validation with the constructed classifier.
Finally we instruct the cross-validation to run on a the loaded data. By default a 10-fold cross validation will be performed and the result for each class will be returned in a Map that maps each class label to its corresponding PerformanceMeasure.

Using the same folds for multiple runs

```
. /* Load data */
. Dataset data = FileHandler.loadDataset(new File [1]("devtools/data/iris.data"), 4
. /* Construct KNN classifier */
. Classifier knn = new KNearestNeighbors(5);
. /* Construct new cross validation instance with the KNN classifier, */
. CrossValidation cv = new CrossValidation(knn);
. /* 5-fold CV with fixed random generator */
. Map<Object, PerformanceMeasure> p = cv.crossValidation(data, 5, new Random [10](1)
. Map<Object, PerformanceMeasure> q = cv.crossValidation(data, 5, new Random [10](1)
. Map<Object, PerformanceMeasure> r = cv.crossValidation(data, 5, new Random [10](2
```

[Documented source code] [11]

The example above performs three rounds of cross-validation on the data set. The first two are using exactly the same folds as the random generator used to create the folds is initialized with the same seed. The third CV will be run on different folds as it uses a different seed.

While in this example we have used the same classifier, one can exchange the classifier with a different one and test different classifiers on exactly the same folds.

Weka classifier

Classification algorithms from Weka can be accessed from within Java-ML and used the same way as the native algorithms by using the WekaClassification bridge. This class can be wrapped around Weka classifiers and makes them transparently available to Java-ML based programs.

In the example below, we first load the iris data set. Next, we create a SMO support vector machine from Weka with default settings. Then, we wrap the SMO in the WekaClassifier bridge. Finally, we perform cross-validation on the classifier and write out the results.

```
. /* Load data */
. Dataset data = FileHandler.loadDataset(new File [1]("iris.data"), 4, ",");
. /* Create Weka classifier */
. SMO smo = new SMO();
. /* Wrap Weka classifier in bridge */
. Classifier javamlsmo = new WekaClassifier(smo);
. /* Initialize cross-validation */
. CrossValidation cv = new CrossValidation(javamlsmo);
. /* Perform cross-validation */
. Map<Object, PerformanceMeasure> pm = cv.crossValidation(data);
. /* Output results */
. System [5].out.println(pm);
```

[Documented source code] [12]

Source URL (retrieved on *07/01/2009 - 04:02*): http://java-ml.sourceforge.net/content/classification

Links:
[1] http://www.google.com/search?hl=en&q=allinurl:file java.sun.com&btnI=I'm Feeling Lucky
[2] http://java-ml.sourceforge.net/src/tutorials/classification/TutorialKNN.java
[3] http://www.google.com/search?hl=en&q=allinurl:object java.sun.com&btnI=I'm Feeling Lucky
[4] http://java-ml.sourceforge.net/content/classification-cross-validation
[5] http://www.google.com/search?hl=en&q=allinurl:system java.sun.com&btnI=I'm Feeling Lucky
[6] http://java-ml.sourceforge.net/src/tutorials/classification/TutorialEvaluateDataset.java
[7] http://java-ml.sourceforge.net/content/load-data-file
[8] http://java-ml.sourceforge.net/content/classification-basics
[9] http://java-ml.sourceforge.net/src/tutorials/classification/TutorialCrossValidation.java
[10] http://www.google.com/search?hl=en&q=allinurl:random java.sun.com&btnI=I'm Feeling Lucky
[11] http://java-ml.sourceforge.net/src/tutorials/classification/TutorialCVSameFolds.java
[12] http://java-ml.sourceforge.net/src/tutorials/tools/TutorialWekaClassifier.java