Orange: From Experimental Machine Learning to Interactive Data Mining

Janez Demsar[1], Blaz Zupan[1,2], Gregor Leban[1] and Tomaz Curk[1]

[1] Faculty of Computer and Information Science, University of Ljubljana, Slovenia
[2] Dep. of Molecular and Human Genetics, Baylor College of Medicine, Houston, USA

Abstract. Orange (`www.ailab.si/orange`) is a suite for machine learning and data mining. It can be used though scripting in Python or with visual programming in Orange Canvas using GUI components called widgets. In the demonstration we will show how to easily prototype state-of-the-art machine learning algorithms through Orange scripting, and design powerful and interactive data exploration applications through visual programming.

1 Orange, a Component-Based Framework

Orange is a comprehensive, component-based framework for machine learning and data mining. It is intended for both experienced users and researchers in machine learning who want to prototype new algorithms while reusing as much of the code as possible, and for those just entering the field who can either write short Python scripts for data analysis or enjoy in the powerful while easy-to-use visual programming environment. Orange includes a range of techniques, such as data management and preprocessing, supervised and unsupervised learning, performance analysis, and a range of data and model visualization techniques.

1.1 Scripting in Python

As a framework, Orange is comprised of several layers. The core design principle was to use C++ to code the basic data representation and manipulation and all time-complex procedures, such as most learning algorithms and data preprocessing. Tasks that are less time consuming are coded in Python. Python is a popular object-oriented scripting language known for its simplicity and power, and often used for prototyping and as a "glue-language" for components written in other languages. The interface between C++ and Python provides a tight integration: Python scripts can access and manipulate Orange objects as if they were implemented in Python. On the other hand, components defined in Python can be used by the C++ core. For instance, one can use classification tree as implemented within Orange (in C++) but prototype a component for attribute selection in Python. For Orange, we took special care to implement machine learning methods so that they are assembled from a set of reusable components

one can either use in the new algorithms, or replace them with prototypes written in Python.

Just for a taste, here is a simple Python script, which, using Orange, reads the data, reports on the number of instances and attributes, builds two classifiers and outputs predicted and true class of the first five instances.

```
import orange
data = orange.ExampleTable('voting.tab')
print 'Instances:', len(data), 'Attributes:', len(data.domain.attributes)
nbc = orange.BayesLearner(data)
knn = orange.kNNLearner(data, k=10)
for i in range(5):
    print nbc(data[i]), knn(data[i]), 'vs. true class', data[i].getClass()
```

Another, a bit more complicated script below, implements a classification tree learner where node attributes that split the data are chosen at random by a function `randomChoice`, which is used in place of data splitting component of Orange's classification tree inducer. The script builds a standard and random tree from the data, and reports on their sizes.

```
import orange, random
def randomChoice(instances, *args):
    attr = random.choice(instances.domain.attributes)
    cl = orange.ClassifierFromVar(whichVar=attr, classVar=attr)
    return cl, attr.values, None, 1

treeLearner = orange.TreeLearner()
rndLearner = orange.TreeLearner()
rndLearner.split = randomChoice

data = orange.ExampleTable('voting.tab')
tree = treeLearner(data)
rndtree = rndLearner(data)
print tree.treesize(), 'vs.', rndtree.treesize()
```

1.2 Visual Programming

Component-based approach was also used for graphical user's interface (GUI). Orange's GUI is made of widgets, which are essentially a GUI wrappers around data analysis algorithms implemented in Orange and Python. Widgets communicate through channels, and a particular set of connected widgets is called a schema. Orange schemas can be either set in Python scripts, or, preferably, are put together by means of visual programming in an application called Orange Canvas.

Besides ease-of-use and flexibility, data exploration widgets were carefully design to support interaction. Clicking on a classification tree node in the tree visualization widget, for example, outputs the corresponding data instances making them available for further analysis. Any visualization of predictive or visualization models where their elements are associated with particular subsets of instances, attributes, data domains, etc., behave in the similar way. A snapshot of Orange Canvas with an example schema is shown in Fig. 1.

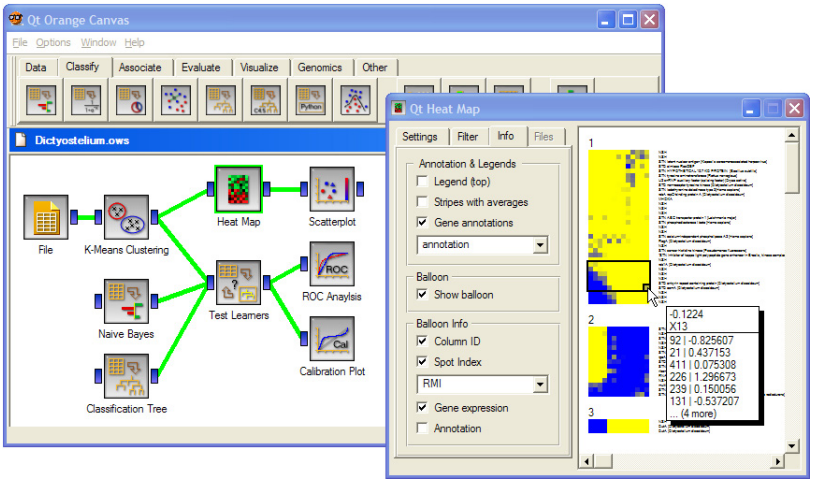

Fig. 1. Snapshot of Orange Canvas with a schema that takes a microarray data, performs k-means clustering, and evaluates the performance of two different supervised learning methods when predicting the cluster label. Clustered data is visualized in the Heat Map widget, which sends any selected data subset to the Scatterplot widget.

2 Orange Demonstration at ECML-PKDD

The demonstration at ECML-PKDD will feature an introduction to Orange, give a tutorial on how to do machine learning by scripting, and show how to use component-based programming to design a new induction algorithm. The second part of the demonstration will demonstrate Orange widgets and visual programming, and will show hot to use these to build classifiers, find interaction between attributes, and do a range of interesting data visualizations. Demonstration will be live, *e.g.* we will write scripts and visually program at the stage rather than show transparencies. The demonstration will use the data from medicine (predictive models) and functional genomics (data mining of microarray and sequence data). CD's with installations and copies of the white-papers will be handed out.

3 On Significance and Contribution

Orange is an open-source framework that features both scripting and visual programming. Because of component-based design in C++ and integration with Python, Orange should appeal to machine learning researchers for the speed of execution and ease of prototyping of new methods. Graphical user's interface is provided through visual programming and carefully designed widgets that support interactive data exploration. Component-based design, both on the level of procedural and visual programming, flexibility in combining components to design new machine learning methods and data mining applications, and user and developer-friendly environment are also the most significant attributes of Orange and those where Orange can make its contribution to the community.