Orange: From Experimental Machine Learning to Interactive Data Mining

Janez Demšar[1], Blaž 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. For researchers in machine learning, Orange offers scripting to easily prototype new algorithms and experimental procedures. For explorative data analysis, it provides a visual programming framework with emphasis on interactions and creative combinations of visual components.

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 write Python scripts to prototype new algorithms while reusing as much of the code as possible, and for those just entering the field who can enjoy in the powerful while easy-to-use visual programming environment. Orange supports various tasks spanning from data preprocessing to modelling and evaluation, such as:

- data management and preprocessing, like sampling, filtering, scaling, discretization, construction of new attributes, and alike,
- induction of classification and regression models, including trees, naive Bayesian classifier, instance-based approaches, linear and logistic regression, and support vector machines,
- various wrappers, like those for calibration of probability predictions of classification models, and those for boosting and bagging,
- descriptive methods like association rules and clustering,
- methods for evaluation and scoring of prediction models, including different hold-out schemes and range of scoring methods and visualization approaches.

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,

J.-F. Boulicaut et al. (Eds.): PKDD 2004, LNAI 3202, pp. 537–539, 2004.

and often used 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.

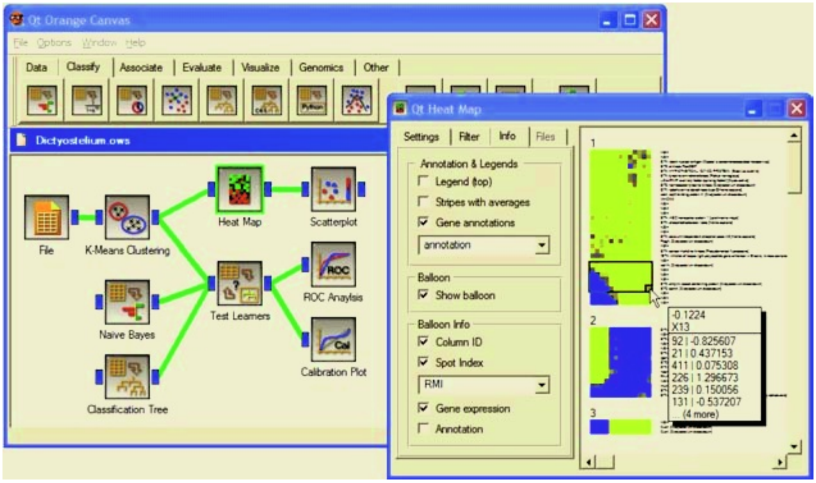
```
import orange, random
def randomChoice(instances, *args):
    attr = random.choice(instances.domain.attributes)
    cl = orange.ClassifierFromVar(whichVar=attr, classVar=attr)
    return cl, attr.values, None, 1

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, designed through visual programming in an application called Orange Canvas.



Annotation & Legends
☐ Legend (top)
☐ Stripes with averages
☑ Gene annotations
annotation

Balloon
☑ Show balloon

Balloon Info
☑ Column ID
☑ Spot Index
RMI
☑ Gene expression
☐ Annotation

-0.1224
X13
92 | -0.825607
21 | 0.437153
411 | 0.075308
226 | 1.296673
239 | 0.150056
131 | -0.537207
... (4 more)

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.

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.

2 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.

Acknowledgement

This work was supported, in part, by the program and project grants from Slovene Ministry of Science and Technology and Slovene Ministry of Information Society, and American Cancer Society project grant RPG-00-202-01-CCE.