How to run an experiment?

September 25, 2007

This tutorial describes various aspects of an experiment using Cougar^2 framework. Especially we will concentrate on experiments using Region Discovery Framework.

To create your experiment create a package in examples.experiment package in the folder src/examples/java. Add your datasets, dataset specification files and experiment files to this package.

1 Datasets

An experiment will require several datasets. All the datasets have to be converted to Cougar^2 data format. Easiest format to use is CSV (Comma Separated Value). Using a software like Excel, you can easily convert a dataset to CSV format. A sample dataset is shown in Table 1. The dataset has three attributes: "X", "Y" and "Region". Here "X" and "Y" are data attributes and "Region" is a meta attribute. It means that "Region" is not used by an algorithm itself, but an algorithm assigns value of it and later we can use this attribute to evaluate an algorithm.

Our dataset is in CSV format with data and meta attributes. To convert it to Cougar^2 format, we will use an XML Dataset specification (spec) file. This XML file will contain details about attributes like, what is the type of attribute, the range of values an attribute can take, whether it is a data or meta attribute etc. Details about the specification can be found in the Javadoc documentation

Table 1: Sample Dataset with two data attributes X, Y and a meta-attribute Region

X	Y	Region
0.02	3.2	0
1.5	0.98	0
2.34	2.21	0
0.78	0.37	0
0.06	1.59	0

Algorithm 1 XML Dataset Spefication for the sample dataset

```
<?xml version="1.0" encoding="UTF-8"?>
<DatasetSpecification>
<Dataset classname="dmml.datasets.numeric.impl.DefaultNumericDataset"/>
<AttributeSpecification>
<DataAttributes>
   <!–X–><AttributeFactory classname="NumericAttrFactory"/>
   <!–Y–><AttributeFactory classname="NumericAttrFactory"/>
</DataAttributes>
<MetaAttributes>
   <!–Region–>
   <AttributeFactory classname="NominalAttrFactory">
   <values type="list">
       <item value="R0" />
       <item value="R1" />
       <item value="R2" />
   </values>
   </AttributeFactory>
</MetaAttributes>
</AttributeSpecification>
</DatasetSpecification>
```

Table 2: Sample Dataset with added meta attributes

X	Y	Region	Reward	Variance
0.02	3.2	0	0	0
1.5	0.98	0	0	0
2.34	2.21	0	0	0
0.78	0.37	0	0	0
0.06	1.59	0	0	0

for DatasetSpecification class[1]. Algorithm 1 shows the specification for sample dataset.

When you want to evaluate a clustering, you may add more meta attributes which will hold values after the evaluation. The example dataset can be extented to hold values like region reward, region interestingness or statistical values like region variance, region Mean Squared Error etc. To append these meta attributes to a dataset, add extra columns to the dataset and initialize them to some value. Update the specification file accordingly. Use this extended dataset while evaluating a clustering result. Table 2 shows the sample dataset with two added meta attributes region reward and region entropy. These columns are initialized to 0 and will be set by the evaluation experiment.

To convert the dataset in the Cougar^2 format, we will use CSVDataset-

[1]`http://cougarsquared.sourceforge.net/build/doc/index.html`

Algorithm 2 How to create a Dataset from a CSV file

```
_dataFileName = TestUtils.packageFileName("myDataset.csv", getClass());
_dataspecFileName = TestUtils.packageFileName("MyDataSpec.xml", getClass());
File dataFile = new File(_dataFileName);
File xmlFile = new File(_dataspecFileName);
CSVDatasetSpecDatasetFactory factory = new CSVDatasetSpecDatasetFactory();
factory.setDataFile(dataFile);
factory.setDatasetSpecFile(xmlFile);
Dataset tData = factory.create();
```

Algorithm 3 How to get a region from a clustering result

```
// rRegion - region label
// rData - Dataset
// rAtt - Region Attribute
public NumericDataset getRegion(int rRegion, NumericDataset rData,
 NominalAttribute rAtt)
{
NumericDataset[] regions;
NominalDatasetUtils nominalDatasetUtils = new DefaultNominalDatasetUtils();
regions = nominalDatasetUtils.split(rData, rAtt);
NumericDataset uRegion = regions[rRegion];
return uRegion;
}
```

SpecDatasetFactory. Algorithm 2 shows how to create a dataset.

2 Getting Regions from an existing clustering

Here we assume that we have run a clustering algorithm on a dataset, i.e. we already have region attribute set for the dataset and we want to evaluate the clustering. If you are using a region discovery algorithm, the algorithm will return each region given its region label/cluster label/cluster id. If the clustering is result of an algorithm outside framework, you will have to write code to get each region. Code in the algorithm 3 will give a region from the dataset given region label.

3 Fitness function and Interestingness Measure

We will use a fitness function and an interstingness measure with a region discovery algorithm and also to evaluate a clustering result. We will use AdditiveFitnessFunction of the Framework. You can use various interestingness measures

Algorithm 4 How to use fitness function and interestingness measure

```
//Interstingness measure to be used
//After creating an object of the interestingness measure,
//you will also configure all the parameters used by it
ExampleMeasure tMeasure = new ExampleMeasure();
...
//Utility classes used by AdditiveFitnessFunction
DefaultNominalDatasetUtils tNominalDatasetUtils =
new DefaultNominalDatasetUtils();
DefaultDatasetUtils tDatasetUtils = new DefaultDatasetUtils();
//Region attribute
NominalAttribute regionAtt =
(NominalAttribute) myData.getMetaAttribute(regionIndex);
double rBeta = 1.01;
AdditiveFitnessFunction tFit = new AdditiveFitnessFunction
(rBeta,tMeasure,regionAtt,tDatasetUtils, tNominalDatasetUtils);
...
//Follwing code assumes that we already have a region from the
//dataset - tRegion
//To get a region reward, we use additive fitness function
tFit.precompute(tRegion);
double reward = tFit.fitness();
//To get a region interestingness, we use the interestingness measure
tMeasure.precompute(tRegion);
double interestingness = tMeasure.fitness();
```

avaialble in the framework. They are in the package : dmml.regionDiscovery.fitness.additive. To know more about additive fitness function and interestingness measures, refere to the Javadoc documentation and the code itself. Algorithm 4 shows how fitness function and interestingness measure will be used to evaluate a region.

4 Running the experiment

Once we are ready with dataset and code, to run an experiment in Cougar^2, use the following steps:

Run > Run > Arguments > Working Directory > Select 'Other' option > Click on 'Workspace' > Select 'bin' folder> Apply > Run.