COSC6335: Data Mining Fall 2008
Project Part II

1. General Picture

In this part of the project, you will conduct some experiments using different clustering algorithms and different datasets. Major project goals are,

- Use the popular K-means and DBSCAN clustering algorithms
- Analyze the results of these algorithms using the Cougar^2 framework and some of its region discovery functions; namely, you will learn what datasets, fitness functions, interestingness measures in the framework are in Part3 you will implement a clustering algorithm yourself.
- Run a region discovery algorithm called CLEVER and SCMRG
- You will use and implement various plug-in fitness functions.
- Visualize and analyze results of running the above four algorithms for 2 datasets.

2. What Will Be Used in the Project?

- Clustering Algorithms: DBSCAN and K-Means (You will use WEKA for these algorithms. Weka can be downloaded from Weka 3.5.6 http://www.cs.waikato.ac.nz/ml/weka/)
- Datasets: Oval10, Earthquake
- Fitness Functions: Purity, Variance, Binary-Co-location; you will implement the binary co-location fitness function yourself, and use the other two fitness functions that already exist in Cougar^2,
- You will also compute the Mean Squared Error (MSE) to evaluate traditional clustering results
- Visualization: Microsoft Excel or Matlab Datasets and a document on 'How to run an Experiment?' can be found at: http://www2.cs.uh.edu/~rachsuda/

3. Project Tasks

3.1 Part A: Traditional Clustering

1. Run the K-means clustering algorithm with K=15, K=10(twice), K=6 for the two datasets tested. Cluster only using the spatial attributes and ignore non-spatial attributes!
2. Run DBSCAN several times for the datasets to determine the best parameter setting for MinPoints and ε. Report the best parameter settings for each dataset. Cluster based on the spatial attributes and ignoring other attributes! Save the best two results you obtained for the two datasets.
3. Visualize the results obtained in steps 1 and 2 (the 4 k-means results, and the two best results obtained by DBSCAN). Interpret the results!
4. Compute the MSE for your best K-means and DBSCAN result. Report the 3 regions that have the lowest MSE for Oval10 and Earthquake.

3.2 Part B: Clustering with Plug-in Fitness Functions

5. Run the CLEVER algorithm with the purity fitness function for the Oval10 dataset (Parameters for CLEVER: $\beta = 1.01$, $\hat{k} = 5$, $p = 60$, $q = 15$, $NeighborhoodSize = 3$, $p\ (insert) = 0.2$, $p\ (delete) = 0.2$, $p\ (replace) = 0.6$ and Parameters for Purity: $\eta = 2$, $th = 0.6$), and visualize and compare the results with those obtained by the other two algorithms in Part1 of the project
6. Implement the Binary Co-location fitness function that is described in Section 6 of the document. Run your fitness function for a set of test cases.
7. Run the CLEVER (Parameters for CLEVER: $\beta = 1.2$, $\hat{k} = 40$, $p = 60$, $q = 15$, $NeighborhoodSize = 3$, $p\ (insert) = 0.2$, $p\ (delete) = 0.2$, $p\ (replace) = 0.6$ and Parameters for Variance: $\eta = 1$, $b = 10$) for the Binary Co-location fitness function you implemented for the Earthquake dataset with respect to earthquake depth and severity attributes. Report and visualize the result!
8. Run the SCMRG and CLEVER (Parameters for CLEVER: $\beta = 1.2$, $\hat{k} = 40$, $p = 60$, $q = 15$, $NeighborhoodSize = 3$, $p\ (insert) = 0.2$, $p\ (delete) = 0.2$, $p\ (replace) = 0.6$) with the variance fitness function ($\eta = 2$, $th = 1.3$) for the Earthquake dataset with respect to the earthquake severity attribute. Analyze and compare and visualize the results.
9. Submit detailed reports that summarize your findings for Part2.

4. Region Discovery Framework

A spatial dataset consists of spatial and non-spatial (meta) attributes. A fitness function that evaluates a clustering depends only on meta attributes and the discovered clustering. The fitness function $q(X)$ depends on a clustering $X = \{r_1, r_2, ..., r_k\}$. Each region $r_i \subseteq O$ is a subset of the dataset O. Regions are mutually exclusive, $r_i \cap r_j = \varnothing$ for $i \neq j$, and mutually exhaustive, $r_1 \cup r_2 \cup ... r_k \subseteq \mathrm{O}$. The clustering and some additional summary information is the output of a region discovery algorithm. We consider a subclass of additive fitness functions:

$$q(X) \quad = \textstyle\sum_{r \in X} q(r) \quad = \displaystyle\sum_{r \in X} i(r)\,|r|^{\beta}$$

where $q(r)$ is the region specific reward function, $|r|$ is the size of the region, and the exponent $\beta > 1$ weights the size of the cluster. The function $i(r)$ is an interestingness measure defined on a region $r \in X$.

5. Design

Figure 1: Class diagram for fitness functions and interestingness measures in Region Discovery Framework

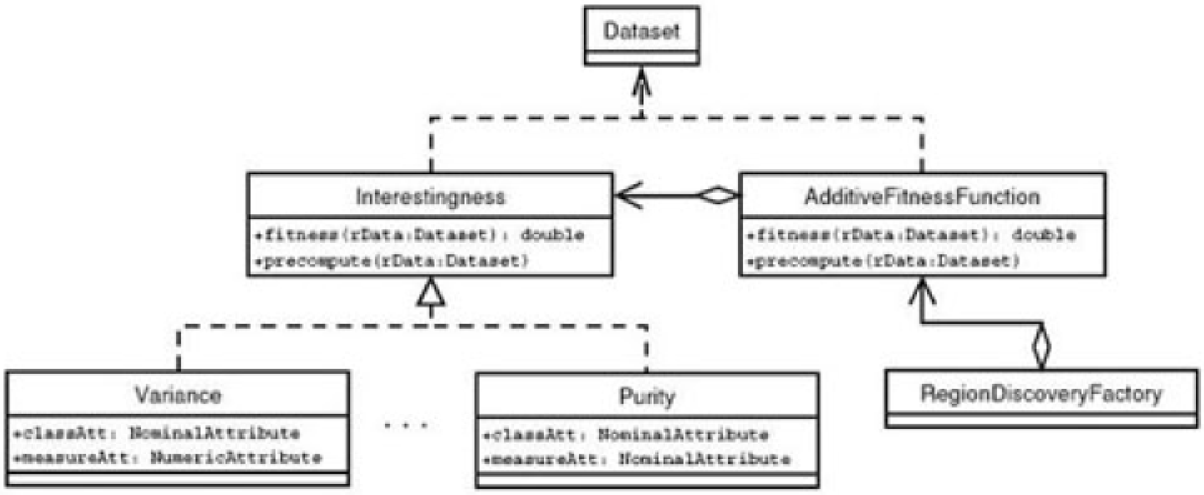
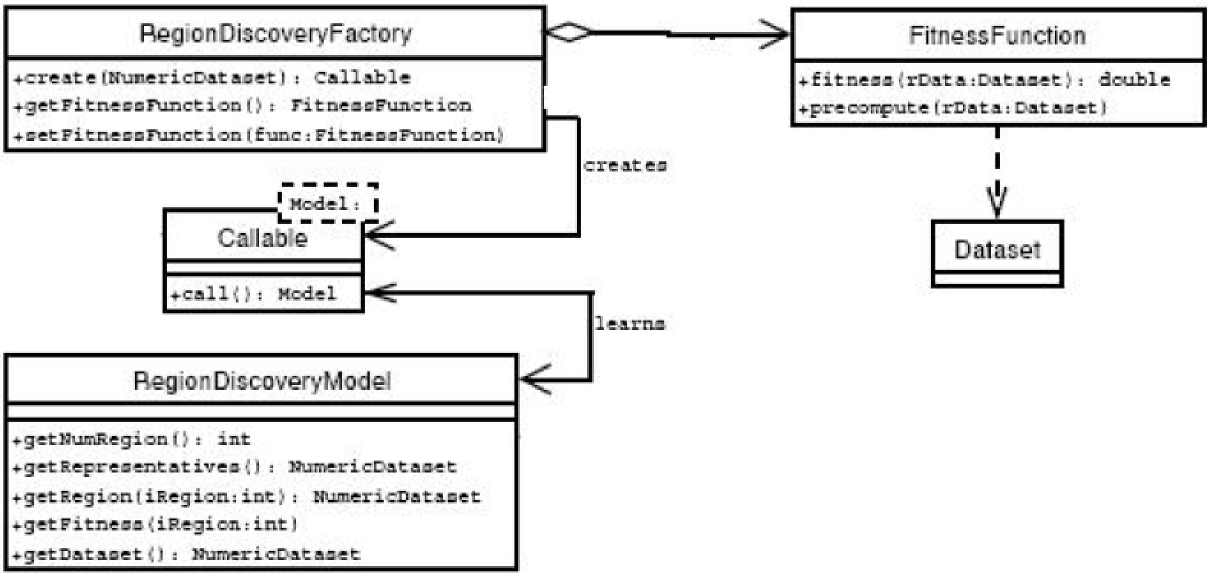

Dataset
Interestingness
+fitness(rData:Dataset): double
+precompute(rData:Dataset)
AdditiveFitnessFunction
+fitness(rData:Dataset): double
+precompute(rData:Dataset)
Variance
+classAtt: NominalAttribute
+measureAtt: NumericAttribute
. . .
Purity
+classAtt: NominalAttribute
+measureAtt: NominalAttribute
RegionDiscoveryFactory

Figure 2: Class diagram for Region Discovery Algorithm classes


RegionDiscoveryFactory
+create(NumericDataset): Callable
+getFitnessFunction(): FitnessFunction
+setFitnessFunction(func:FitnessFunction)
FitnessFunction
+fitness(rData:Dataset): double
+precompute(rData:Dataset)
creates
Model:
Callable
+call(): Model
Dataset
learns
RegionDiscoveryModel
+getNumRegion(): int
+getRepresentatives(): NumericDataset
+getRegion(iRegion:int): NumericDataset
+getFitness(iRegion:int)
+getDataset(): NumericDataset

6. Measures of Interestingness and Error Measures

For simplicity we assume that the argument is a Dataset object and this Dataset corresponds to a single region.

6.1 Mean Squared Error (MSE)

Let μ be the mean of the values in the dataset. The measure of interestingness is defined as:

$$\begin{aligned} MSE(r) &= \frac{1}{m}\sum_{o\in r}\|r - centroid(r)\|^2 \\ centroid(r) &= \frac{1}{m}\sum_{o\in r} r \end{aligned}$$

In this measure, only the real data is used and not the meta-data. All data attributes are numeric attributes. $\eta > 1$ is the scaling factor. $th > 0$ is the threshold. m is the size of the dataset. Euclidian distance is used for the distance calculation.

6.2 Purity

Purity is defined as fraction of examples that belong to the majority class. The majority class is the class with the most number of examples. In the event that two classes have an equal number of examples, the majority class is arbitrarily selected.

$$i(r) = \begin{cases} 0 & \max_c p_c(r) < th \\ (\max_c p_c(r) - th)^\eta & otherwise \end{cases}, \ c \in cl(O)$$

Attribute used is the nominal class attribute. $\eta > 1$ is the scaling factor. $cl(O)$ is the set of classes in the dataset. $th > 0$ is the threshold.

6.3 High Variance Interestingness

Let

th≥1 be the reward threshold[1]

∞>η>0 be the reward function form parameter[2]

Then the interestingness of a region *r*, denoted by *i*(*r*), is defined as follows:

$$i(r) = \begin{cases} 0 & \dfrac{Var(r(m))}{Var(O(m))} \leq th \\ \left(\dfrac{Var(r(m))}{Var(O(m))} - th \right)^{\eta} & otherwise \end{cases}$$

where

$$Var(r(m)) = \frac{1}{|r| - 1} \sum_{o \in r} (o.x_m - \mu_m)^2$$

where n is the number of examples, x_m is the value of the mth meta-attribute in x, and μ is the mean value of the mth meta-attribute. $b > 1$ is the base of the logarithm. $\eta > 0$ is the scaling factor. $O\ (m)$ is the mth meta-attribute of the dataset and $Var\ (O\ (m))$ is the variance of the mth meta-attribute in the dataset.

6.4 Hotspot Interestingness

Let

th≥1 be the reward threshold[3]

∞>η>0 be the reward function form parameter[4]

Then the interestingness of a region r, denoted by *i*(*r*), is defined as follows:

$$i(r) = \begin{cases} 0 & Avg(r(m)) \leq th \\ (Avg(r(m)) - th)^{\eta} & otherwise \end{cases}$$

where

$$Avg(r(m)) = \frac{1}{|r|} \sum_{o \in r} O.x_m$$

6.5 Co-location Interestingness

We also interest in regions where high magnitude and deep earthquake are co-located. The following is definition of Co-location Interestingness.

Given a set of continuous attributes *A* = {*A*$_1$,…, *A*$_q$} the interestingness of an object *o*∈*O* is measured as follows:

[1] It determines how much higher the region variance has to be than the data set variance to obtain a reward.

[2] For example, using η=3 defines cubic reward function, whereas η=1 implies that increases in the ratio between the region variance and the dataset variance are rewarded linearly.

[3] It determines how high the region average has to be to obtain a reward.

[4] For example, using η=3 defines cubic reward function, whereas η=1 implies that increases in the region average is rewarded linearly.

$$i(A,o) = \prod_{j=1}^{q} z_{A_j}(o)$$

where z_A denotes the z-score of the continuous attributes *A*.

The definition of interestingness of an object is then extended to the definition of interestingness of a region. The interestingness of a region *r* is the absolute value of the average interestingness of the objects belonging to it:

$$i(A,r) = \left\{ \begin{matrix} \left(\dfrac{\left|\sum_{o\in r} i(A,o)\right|}{size(r)} - th \right)^{\eta} & if\ \dfrac{\left|\sum_{o\in r} i(A,o)\right|}{size(r)} > th \\ 0 & otherwise \end{matrix} \right\}$$

6 Coding Conventions

The code that you will write for the experiments should follow a few programming conventions:
_X X is a private member variable.
iX is an index to the array or collection named X.
tY Y is the value you are testing as in: tSize = _testObject.size();
eX X is the known value against which you are comparing as in: eSize = 7;
assertEquals(eSize, tSize);
fX is an element of the array named X whose index is iX as in: fX = X[iX];
uX X is the return value of an method as in: uSum = 1+2; return uSum;
nX length of the array X as in: nX = X.length; or nX = X.size(); or nX = X.getNumMetaAttributes();
rX Read-only reference to X in method signature
wX Read-Write reference to X in method signature
whatXDoes The name of a variable should indicate how it is used in the function. Avoid single letter names as in (bad): z = x[y] could better be written fCluster = cluster[iCluster]

7 Submission

Notice different deadlines for the different tasks of the project:
1. Submit a hard copy of your results at PGH577 for Part A by October 8, 11:00 PM CST.
2. Email your code to rachsuda@cs.uh.edu and submit a hard copy of the detailed report for Part B at PGH577 by October 22, 11:00 PM CST.