Documentation for the Data Complexity Library in C++

Albert Orriols-Puig, Núria Macià, and Tin Kam Ho

GRSI Report No. 2010001
December 2010

Grup de Recerca en Sistemes Intel·ligents
La Salle - Universitat Ramon Llull
C/ Quatre Camins, 2
08022, Barcelona (Spain)

Contents

Documentation for the Data Complexity Library in C++

Albert Orriols-Puig[1], Núria Macià[1], and Tin Kam Ho[2]

[1]Grup de Recerca en Sistemes Intel·ligents
La Salle - Universitat Ramon Llull
C/ Quatre Camins, 2. 08022, Barcelona (Spain)
{aorriols,nmacia}@salle.url.edu

[2]Statistics and Learning Research Group
Bell Laboratories, Alcatel-Lucent
Murray Hill, NJ 07974-0636 USA
tkh@research.bell-labs.com

Abstract

This report supplies documentation for the C++ implementation of the data complexity library, which provides a set of measures that evaluate the geometrical complexity of classification problems. More specifically, the implemented measures focus on the complexity of the class boundary and estimate (1) the overlaps in the feature values from different classes, (2) the class separability, and (3) the geometry, topology, and density of manifolds. In addition, two other complementary functionalities, (4) stratified k-fold partitioning and (5) routines to transform m-class data sets ($m > 2$) into m two-class data sets, are included in the library. Lastly, the source code can be compiled across multiple platforms and can be easily configured and run from the command line.

1 Introduction

This report is the documentation for the *data complexity library*[1] (DCoL). This library provides the implementation of a set of measures designed to characterize the apparent complexity of data sets for supervised learning, which were originally proposed by Ho and Basu (2002). More specifically, the following measures have been implemented:

- *Measures of overlaps in the feature values from different classes.* This library provides routines that compute (1) the maximum Fisher's discriminant ratio in both its original form defined by (Ho and Basu, 2002) and its directional-vector form (see the details in Section 2), (2) the overlap of the per-class bounding boxes, (3) the maximum (individual) feature efficiency, and (4) the collective feature efficiency[2].
- *Measures of class separability.* This library provides routines that compute (1) the minimized sum of the error distance of a linear classifier, (2) the training error of a linear classifier. (3) the fraction of points on the class boundary, (4) the ratio of average intra/inter class nearest neighbor distance, and (5) the leave-one-out error rate of the one-nearest neighbor classifier.

[1]The software release is available at http://dcol.sourceforge.net.

[2]The collective feature efficiency is a new complexity measure based on the individual feature efficiency measure proposed by Ho and Basu (2002).

- *Measures of geometry, topology, and density of manifolds.* This library provides routines that compute (1) the nonlinearity of a linear classifier, (2) the nonlinearity of the one-nearest neighbor classifier, (3) the fraction of maximum covering spheres, and (4) the average number of points per dimension.

The implementation of these complexity measures is based on the descriptions provided by Ho and Basu (2002) and Ho et al. (2006). The majority of these measures were initially designed for two-class data sets and were only applied to problems with continuous attributes[3] (nominal or categorical attributes were numerically coded and treated as continuous). The latter restriction was because most of the complexity measures rely on distance functions between attributes. In our implementation, all the measures except for those based on linear discriminants and the Fisher's discriminant in its directional-vector have been extended to deal with m-class data sets ($m > 2$), following the guidelines suggested by Ho et al. (2006). Furthermore, the most relevant distance functions for continuous and nominal attributes (Wilson and Martinez, 1997) have been implemented. In this way, we enable our library to deal with m-class data sets that contain nominal and/or continuous attributes.

The library also offers two other functionalities:

- Partition of the data set by means of stratified k-fold cross-validation (Dietterich, 1998).
- Conversion of an m-class data set ($m > 2$) to m two-class data sets. Each new data set discriminates one of the classes against the others.

Finally, in order to maintain the standard of open source, the code was carefully tested on a large collection of real-world problems selected from the UCI repository (Asuncion and Newman, 2007) (see the details in Appendix B).

The remainder of this report is structured as follows. Section 2 highlights the importance of characterizing the complexity of classification problems to get a better understanding of our learners and describes the complexity measures that have been implemented. Next, Section 3 provides information on how to download the library, extract the source, and compile, run and call the code. Section 4 shows some examples of runs on a set of real-world classification problems. Finally, Section 5 reviews the code structure, and Section 6 provides the user with some indications on how to extend the code with new complexity measures.

2 Data Complexity in Supervised Learning

The complexity of classification problems (Basu and Ho, 2006) has been traditionally attributed to three main sources: (1) *class ambiguity*, (2) *boundary complexity*, and (3) *sample sparsity and feature space dimensionality* (Ho and Basu, 2002; Ho et al., 2006). In the following, these three sources of data complexity are discussed in some detail.

Class ambiguity refers to the situation in which examples of different classes cannot be distinguished by the problem features. This could be due to a poor capability of the selected attributes to describe the concepts that (1) belong to different classes (i.e., the attributes of the problem are not sufficient to describe the concepts) or (2) belong to classes that are not well defined or have

[3]In this document, we use the terms attribute and feature indistinctively to refer to each feature that describes a problem instance. We also use without distinction the terms instance and example.

some relationship among them (e.g., having some instances that belong to two classes). This type of complexity cannot be solved at the classifier level, and data preprocessing may be needed to disambiguate the classes or the concepts. Data sets that contain classes that are ambiguous for some cases are said to have nonzero Bayes error, which sets a lower bound on the achievable error rate.

Boundary complexity is related to the length of the description needed to describe the class. Given a complete sample, the Kolmogorov complexity (Kolmogorov, 1965; Li and Vitanyi, 1993) is defined as the length of the shortest program that describes the class boundary. Nonetheless, the Kolmogorov complexity is known to be incomputable (Maciejowski, 1979). Therefore, other estimates have been designed to analyze the class complexity, which mainly extract different *geometrical indicators* from the data set. Note, moreover, that the boundary complexity is closely related to the knowledge representation used by the learners. Thence, the type of representation used may impose a minimum bound of the classification error. For example, linear classifiers can hardly fit curved boundaries, and so, they would accumulate large errors on the class boundary; conversely, kernel-based methods could easily reproduce curved boundaries if the kernel has enough freedom to fit the boundary shape.

Finally, *sample sparsity and feature space dimensionality* searches for characterizing complexities generated by regions with sparse samples in the feature space. Generalization over empty spaces of the training data set is largely arbitrary and depends mainly on how the classifier constructs the data model. The difficulty of dealing with sparse samples in high dimensional spaces have been addressed in many works (Devroye, 1988; Raudys and Jain, 1991; Vapnik, 1998), and some approaches expressly avoid evolving knowledge in empty regions in the feature space (Casillas et al., 2008).

Among the different sources of problem difficulties, boundary complexity has received especial attention since it is the type of complexity that is more likely to be assessed. In particular, Ho and Basu (2002) designed a set of measures that extract different indicators that characterize the apparent geometrical complexity of the problem boundary. These measures can be divided into the following three categories:

Measures of overlaps in the feature values from different classes. These measures focus on the capacity of the features to separate examples of different classes. For each individual attribute, they examine the range and spread of the values of instances of different classes and check the discriminant power of a single attribute or a combination of them. The library offers the following measures: (1) the maximum Fisher's discriminant ratio (F1), (2) the overlap of the per-class bounding boxes (F2), and (3) the maximum (individual) feature efficiency (F3). In addition, we designed two extra measures based on the previous ones: (4) the directional-vector maximum Fisher's discriminant ratio (F1v), inspired by F1, and (5) the collective feature efficiency (F4), inspired by F3.

Measures of class separability. These measures estimate to what extent the classes are separable by examining the length and the linearity of the class boundary. The library offers the following measures: (1) the minimized sum of the error distance of a linear classifier (L1), (2) the training error of a linear classifier (L2), (3) the fraction of points on the class boundary (N1), (4) the ratio of average intra/inter class nearest neighbor distance (N2), and (5) the leave-one-out error rate of the one-nearest neighbor classifier (N3).

Measures of geometry, topology, and density of manifolds. These measures provide an indirect characterization of the class separability. They assume that the problem is composed of several manifolds spanned by each class. The shape, position, and interconnectedness of these manifolds

give some hints on how well the classes are separated and on the density or population of each manifold. The library offers the following measures: (1) the nonlinearity of a linear classifier (L3), (2) the nonlinearity of the one-nearest neighbor classifier (N4), (3) the fraction of maximum covering spheres (T1), and (4) the average number of points per dimension (T2).

These complexity measures are explained in more detail in the next subsection. For further information the reader is referred to (Ho and Basu, 2002; Ho et al., 2006). For implementation details, please see the code.

2.1 Measures of Overlaps in the Feature Values from Different Classes

We first start with the description of the five measures that estimate different complexities related to discriminative power of the attributes.

Maximum Fisher's discriminant ratio (F1). This measure computes the maximum discriminative power of each attribute, that is,

$$F1 = \max_{a=1}^{\ell} f_a, \tag{1}$$

where ℓ is the number of input attributes, and f_a is the discrimant ratio of each attribute. f_a is computed differently depending on whether the data set has two classes or more than two classes.

1. For two-class data sets, the ratio for each attribute a is computed as

$$f_a = \frac{(\mu^a_{c_1} - \mu^a_{c_2})^2}{(\sigma^a_{c_1})^2 + (\sigma^a_{c_2})^2}, \tag{2}$$

   where, for continuous attributes, $\mu^a_{c_i}$ and $(\sigma^a_{c_i})^2$ are the mean and the variance of the attribute a for class c_i. For nominal attributes, each value is mapped into an integer number. Then, $\mu^a_{c_i}$ is the median value of the attribute a for class c_i, and $(\sigma^a_{c_i})^2$ is the variance of the attribute a for class c_i computed as the variance of the binominal distribution, that is,

$$\sigma^a_{c_i} = \sqrt{p_{\mu^a_{c_i}}(1 - p_{\mu^a_{c_i}}) * n_{c_i}}, \tag{3}$$

   where $p_{\mu^a_{c_i}}$ is the frequency of the median value $\mu^a_{c_i}$ and n_{c_i} is the total number of examples of class c_i.

2. For m-class data sets ($m > 2$), the ratio for each attribute a is computed as

$$f_a = \frac{\sum_{c_i=1}^{C} \sum_{c_j=c_i+1}^{C} p_{c_i} p_{c_j} (\mu^a_{c_i} - \mu^a_{c_j})^2}{\sum_{c_i=1}^{C} p_{c_i} \sigma^2_{c_i}}, \tag{4}$$

   where C is the maximum number of classes and p_{c_i} is the proportion of examples of class c_i.

A high value of Fisher's discriminant ratio indicates that, at least, one of the attributes enables the learner to separate the examples of different classes with partitions that are parallel to an axis of the feature space. A low value of this measure does not imply that the classes are not linearly separable, but that they cannot be discriminated by hyperplanes parallel to one of the axis of the feature space.

The directional-vector maximum Fisher's discriminant ratio (F1v). This measure complements F1 by searching for an oriented vector which can separate examples of two different classes.

It computes the two-class Fisher's criterion, which takes the following form (Malina, 2001):

$$R(d) = \frac{[\vec{d}^T(\vec{\mu_1} - \vec{\mu_2})]^2}{\vec{d}^T\overline{\Sigma}\vec{d}} = \frac{\vec{d}^T B\vec{d}}{\vec{d}^T\overline{\Sigma}\vec{d}}, \tag{5}$$

where

- $\vec{d}$ is the directional vector on which data are projected
- $\vec{\mu}_i$ is the mean vector
- $\overline{\Sigma} = a\Sigma_1 + (1-a)\Sigma_2,\ 0 \leq a \leq 1$
- Σ_i is the scatter matrix of patterns for class c_i
- $B = (\vec{\mu}_1 - \vec{\mu}_2)(\vec{\mu}_1 - \vec{\mu}_2)^T$ is the between class scatter matrix

The directional vector $\vec{d}$ is calculated as

$$\vec{d} = \overline{\Sigma}^{-1}\Delta, \tag{6}$$

where $\Delta = \mu_1 - \mu_2$ and $\overline{\Sigma}^{-1}$ is computed as the pseudo inverse (Moore, 1920; Penrose, 1955) of $\overline{\Sigma}$.

This measure is implemented only for two-class problems. A high value of the Fisher's discriminant ratio indicates that there exists a vector that can separate examples belonging to different classes after these instances are projected on it.

The overlap of the per-class bounding boxes (F2). This measure computes the overlap of the tails of distributions defined by the instances of each class.

The definition of this measure for two-class data sets is the following. For each attribute, it computes the ratio of the width of the overlap interval (i.e., the interval that has instances of both classes) to the width of the entire interval. Then, the measure returns the product of the ratios calculated for each attribute:

$$F2 = \prod_{a=1}^{\ell} \frac{\text{MIN\_MAX}_a - \text{MAX\_MIN}_a}{\text{MAX\_MAX}_a - \text{MIN\_MIN}_a}, \tag{7}$$

where ℓ is the number of input attributes, and

$$\text{MIN\_MAX}_a = \min(\max(f_i, c_1), \max(f_i, c_2)), \tag{8}$$

$$\text{MAX\_MIN}_a = \max(\min(f_i, c_1), \min(f_i, c_2)), \tag{9}$$

$$\text{MAX\_MAX}_a = \max(\max(f_i, c_1), \max(f_i, c_2)),\ \text{and} \tag{10}$$

$$\text{MIN\_MIN}_a = \min(\min(f_i, c_1), \min(f_i, c_2)), \tag{11}$$

where f_i is the ith feature, c_1 and c_2 refer to the two classes, and $\max(f_i,c_i)$ and $\min(f_i,c_i)$ are, respectively, the maximum and minimum values of the feature f_i for class c_i. Nominal values are mapped to integer values to compute this measure.

For m-class data sets ($m > 2$), we compute F2 for each pair of classes, get the absolute value of all them, an return the sum of all these values.

A low value of this measure means that the attributes can discriminate the examples of different classes.

The maximum (individual) feature efficiency (F3). This measure computes the discriminative power of individual features and returns the value of the attribute that can discriminate the largest number of training instances.

For this purpose, the following heuristic is employed. For each attribute, we consider the overlapping region (i.e., the region where there are instances of both classes) and return the ratio of the number of instances that are not in this overlapping region to the total number of instances. Then, the maximum discriminative ratio is taken as measure F3.

Note that a problem is easy if there exists one attribute for which the ranges of the values spanned by each class do not overlap (in this case, this would be a linearly separable problem).

The collective feature efficiency (F4). This measure follows the same idea presented by F3, but now it considers the discriminative power of all the attributes (therefore, the *collective* feature efficiency).

To compute the collective discriminative power, we apply the following procedure. First, we select the most discriminative attribute, that is, the attribute that can separate a major number of instances of one class. Then, all the instances that can be discriminated are removed from the data set, and the following most discriminative attribute (regarding the remaining examples) is selected. This procedure is repeated until all the examples are discriminated or all the attributes in the feature space are analyzed. Finally, the measure returns the proportion of instances that have been discriminated. Thus, it gives us an idea of the fraction of examples whose class could be correctly predicted by building separating hyperplanes that are parallel to one of the axis in the feature space.

Note that the measure described herein slightly differs from the *maximum feature efficiency*. Maximum feature efficiency only considers the number of examples discriminated by the most discriminative attribute, instead of all the attributes. Thence, F4 provides more information by taking into account all the attributes since we want to highlight the collective discriminative power of all the features.

2.2 Measures of Class Separability

In what follows, we describe five measures that examine the shape of the class boundary to estimate the complexity of separating instances of different classes.

The minimized sum of the error distance of a linear classifier (L1). This measure evaluates to what extent the training data is linearly separable.

For this purpose, it returns the sum of the difference between the prediction of a linear classifier and the actual class value. Different from Ho and Basu (2002), in our implementation we use a support vector machine (SVM) (Vapnik, 1995) with a linear kernel, which is trained with the *sequential minimal optimization* (SMO) algorithm (Platt, 1998) to build the linear classifier. We used this learner since the SMO algorithm provides an efficient training method, and the result is a linear classifier that separates the instances of two classes by means of a hyperplane.

This measure is implemented only for two-class problems. A zero value of this measure indicates that the problem is linearly separable.

The training error of a linear classifier (L2). This measure provides information about to what extent the training data is linearly separable. It builds the linear classifier as explained above and returns its training error. As before, the measure is only implemented for two-class data sets.

The fraction of points on the class boundary (N1). This measure gives an estimate of the length of the class boundary.

For this purpose, it builds a *minimum spanning tree* over the entire data set and returns the ratio of the number nodes of the spanning tree that are connected and belong to different classes to the total number of examples in the data set. If a node n_i is connected with nodes of different classes, n_i is counted only once.

High values of this measure indicate that the majority of the points lay closely to the class boundary, and so, that it may be more difficult for the learner to define this class boundary accurately.

The ratio of average intra/inter class nearest neighbor distance (N2). This measure compares the within-class spread with the distances to the nearest neighbors of other classes.

For each input instance ex_i, we calculate the distance to its nearest neighbor within the class (intraDist(ex_i)) and the distance to its nearest neighbor of any other class (interDist(ex_i)). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{N_e} intraDist(ex_i)}{\sum_{i=0}^{N_e} interDist(ex_i)}, \tag{12}$$

where N_e is the number of examples in the data set.

Low values of this measure suggest that the examples of the same class lay closely in the feature space. High values indicate that the examples of the same class are disperse.

The leave-one-out error rate of the one-nearest neighbor classifier (N3). The measure denotes how close the examples of different classes are. It returns the leave-one-out error rate of the one-nearest neighbor (the kNN classifier with k=1) learner.

Low values of this measure indicate that there is a large gap in the class boundary.

2.3 Measures of Geometry, Topology, and Density of Manifolds

Having seen a set of measures that estimate the shape of the class boundary, we now explicate four measures that indirectly characterize the class separability by assuming that a class is made up of single and multiple manifolds that form the support of the distribution of the class.

The nonlinearity of a linear classifier (L3). This measure implements a measure of nonlinearity proposed by Hoekstra and Duin (1996).

Given the training data set, the method creates a test set by linear interpolation with random coefficients between pairs of randomly selected instances of the same class. Then, the measure returns the test error rate of the linear classifier (the support vector machine with linear kernel) trained with the original training set. The measure is sensitive to the smoothness of the classifier boundary and the overlap on the convex hull of the classes. This measure is implemented only for two-class data sets.

The nonlinearity of the one-nearest neighbor classifier (N4). This measure creates a test set as proposed by L3 and returns the test error of the 1NN classifier.

The fraction of maximum covering spheres (T1). This measure was originated in the work of Lebourgeois and Emptoz (1996), which described the shapes of class manifolds with the notion of *adherence subset*. Simply speaking, an adherence subset is a sphere centered on an example of the data set which is grown as much as possible before touching any example of another class. Therefore, an adherence subset contains a set of examples of the same class and cannot grow more without including examples of other classes. The measure considers only the biggest adherence subsets or spheres, removing all those that are included in others. Then, the measure returns the number of spheres normalized by the total number of points.

The average number of points per dimension (T2). This measure returns the ratio of the number of examples in the data set to the number of attributes. It is a rough indicator of sparseness of the data set.

In order to increase the functionalities of the software, some additional functions such as automatic transformation of m-class data sets into m two-class data sets and k-fold cross-validation partitioning have been included in the library. Details on how to download, compile, configure, and run the software are supplied in the next section.

3 How to Compile, Run, and Call the Code

The software release composed of the source code, some examples of classification problems, and the documentation (this report) is available at `http://dcol.sourceforge.net`. In the following subsections, we explain the system requirements to run the code and provide a manual on how to compile, run, and specify the input options of the software as well as how to formulate calls of the complexity measures routines from own code.

3.1 System Requirements

The only requirement to run the DCoL is to have a C++ compiler installed in your computer. The code was implemented on a Unix/Linux 32-bit architecture, and *g++*[4] was used. Nonetheless, it does not use third party libraries for compatibility across platforms, so that the code can be compiled and run in any platform that has a C++ compiler. The instructions given in the description below correspond to Unix/Linux commands. The same process can be followed with the equivalent instructions for Windows and Mac operating systems.

3.2 How to Compile the Code

After downloading the software release, the sources can be extracted typing the following commands:

```
gunzip DCoL-v1.1.tar.gz
tar xvf DCoL-v1.1.tar
```

For Windows operating system, please use any available unzip software. This command will create five folders: (1) `Data`, (2) `Source`, (3) `Test`, (4) `Help`, and (5) `Documentation`. `Data` contains data set examples belonging to the UCI repository. Specifically, there are four two-class data sets—*h-s*, *pim*, *tao*, and *wbcd*—, and three m-class data sets ($m > 2$)— *bal*, *iris*, and *thy*. `Source` has the source files of the library (including the folder `DistanceFunctions` which has the implementation of

[4] http://gcc.gnu.org/

the distance functions) and a Makefile. `Test` contains a folder named `Datasets` which includes some data sets used for the testing and the script to replicate the experiments reported in the Appendix B. `Help` has the on-line documentation browser (in HTML) and the off-line reference manual (in LaTeX) from the source files. `Documentation` contains this report as a user and programmer manual.

To compile the code in a Unix/Linux environment, go to the `Source` folder and call `make`. This will generate a set of object files plus the application file, which is named `dcol`. Temporal and object files can be removed with the command `make clean`. For Windows and Mac systems, please follow the compilation procedure of the chosen compiler.

Note: `gettimeofday()` is not a native Windows function. So, please comment out lines 64-69 in file `Utils.cpp` and uncomment lines 72-73 (see Figure 1).

```
63.    // For Linux and MacOS X.
64.    struct timeval tv;
65.    time_t curtime;
66.    gettimeofday ( &tv, NULL );
67.    curtime = tv.tv_sec;
68.    strftime ( buffer, 30, "%T", localtime ( &curtime ) );
69.    sprintf ( buffer, "%s%ld", buffer, ( long int ) tv.tv_usec );
70.
71.    // For Ms Windows.
72.    // srand ( time ( NULL ) );
73.    // sprintf ( buffer, "%d", rand () );
```

Figure 1: Piece of code from file `Utils.cpp`

3.3 How to Run the Code

After compiling the code, the application can be run with the command `./dcol`. The input format, command line options, and output format are specified below.

3.3.1 Format of the Input

The application requires an input file which must contain either (1) the data set for runs with a single data set or (2) a list of paths to data set files for runs with multiple data sets. The input data sets can follow either the WEKA format (Witten and Frank, 2005) or the KEEL format (Alcalá-Fdez et al., 2008) which is an extension of the WEKA format. Appendix A supplies the specification for the KEEL format; for more information, please visit `http://www.keel.es`. Note that the application does not accept string attributes.

The next subsections explain how to perform a single run (with one data set) or multiple runs (with several data sets) and specify the format of the output files.

3.3.2 Command Line Options

As follows, we specify the command line syntax and the different options of the application.

Table 1: Computational cost of the complexity measures. e is the number of input examples, e_t is the number of test examples (applicable only to the measures that generate an additional test set), a is the number of attributes, and c is the number of classes. $O(SMO)$ is the cost of building a support vector machine with linear kernel by means of the sequential minimal optimization algorithm.

Label	Complexity Measure	Computational Cost
F1	Maximum Fisher's discriminant ratio	$O(e \cdot a)$
F1v	Directional-vector maximum Fisher's discriminant ratio	$O(e \cdot a + a^3)$
F2	Overlap of the per-class bounding boxes	$O(e \cdot a)$
F3	Maximum (individual) feature efficiency	$O(e \cdot a \cdot c)$
F4	Collective feature efficiency	$O(e \cdot a^2 \cdot c)$
L1	Minimized sum of the error distance of a linear classifier	$O(SMO)$
L2	Training error of a linear classifier	$O(SMO)$
L3	Nonlinearity of a linear classifier	$O(SMO + e_t \cdot a \cdot c)$
N1	Fraction of points on the class boundary	$O(e^2 \cdot a)$
N2	Ratio of average intra/inter class nearest neighbor distance	$O(e^2 \cdot a)$
N3	Leave-one-out error rate of the one-nearest neighbor classifier	$O(e^2 \cdot a)$
N4	Nonlinearity of the one-nearest neighbor classifier	$O(e_t \cdot a \cdot c + e \cdot e_t \cdot a)$
T1	Fraction of maximum covering spheres	$O(e^2 \cdot a)$
T2	Average number of points per dimension	$O(1)$

Command Line Syntax
The syntax to call the application is

```
./dcol -i <input_file> -o <output_file> [OPTIONS]
```

Two parameters *are required* to run the application:

1. `-i <input_file>`: It specifies the name of the file with either (1) the input data set in KEEL or WEKA format for runs with a single data set (i.e., option `-B` is not specified) or (2) a list of data set paths and names for runs with multiple data sets (if option `-B` is specified).

2. `-o <output_file>`: It sets the name of the output file where the results of the application will be written.

In addition, a set of optional parameters can be specified (i.e., `[OPTIONS]`). In the following, we show the syntax for (1) the options to select which complexity measures are run, (2) the options to select distance functions for continuous attributes and nominal attributes, and (3) other options, which allow for other functionalities such as transformation of m-class data sets ($m > 2$) into m two-class data sets and stratified cross-validation.

Complexity Measures Options
As follows, we provide the options that permit running the complexity measures detailed in Section 2. Please remember that some measures have been slightly modified from their initial definition (Ho and Basu, 2002). Specifically, a support vector machine with a linear kernel is used as linear discriminant in L1, L2, and L3. Moreover, two new measures have been designed: F1v, which corresponds to a vectorial form of the Fisher's discriminant and F4, which computes the collective feature efficiency. Table 1 summarizes the computational cost of each complexity measure.

1. `-F 1`: It computes the *maximum Fisher's discriminant ratio*. Its computational cost grows linearly with the sum of the product of the number of input examples and the number of attributes.

2. `-F 1v`: It computes the *directional-vector maximum Fisher's discriminant ratio*. Its computational cost grows linearly with the sum of (1) the product of the number of input examples and the number of attributes and (2) the number of attributes power three.

3. `-F 2`: It computes the *overlap of the per-class bounding boxes*. Its computational cost grows linearly with the product of the number of input examples and the number of attributes.

4. `-F 3`: It computes the *maximum (individual) feature efficiency*. Its computational cost grows with the product of the number of examples, the number of attributes, and the number of classes.

5. `-F 4`: It computes the *collective feature efficiency*. Its computational cost grows with the product of the number of examples, the number of attributes power two, and the number of classes.

6. `-L 1`: It computes the *sum of the error distance of a linear classifier*. Its computational cost is guided by the cost of building the support vector machine (for further details, see (Platt, 1998)).

7. `-L 2`: It computes the *training error of a linear classifier*. Again, the computational cost is guided by the cost of building the support vector machine (for further details, see (Platt, 1998)).

8. `-L 3`: It computes the *nonlinearity of a linear classifier*. Its computational cost is guided by the cost of building the support vector machine (see (Platt, 1998)) plus the cost of generating the test set which increases linearly with the number of test examples, the number of classes, and the number of attributes.

9. `-N 1`: It computes the *fraction of points on the class boundary*. Its computational cost is determined by the cost of building the minimum spanning tree, which grows with the product of the number of examples power two and the number of attributes.

10. `-N 2`: It computes the *ratio of average intra/inter class nearest neighbor distance*. Its computational cost depends on the product of the number of examples power two and the number of attributes.

11. `-N 3`: It computes the *leave-one-out error rate of the one-nearest neighbor classifier*. Its computational cost depends linearly on the product of the number of examples power two and the number of attributes.

12. `-N 4`: It computes the *nonlinearity of the one-nearest neighbor classifier*. Its computational cost depends on the sum of (1) the cost of generating the test examples and (2) the cost of computing the kNN algorithm. The former one increases linearly with the product of the number of test examples that are generated, the number of attributes, and the number of classes. The latter one depends linearly on the product of the number of training examples, the number of test examples, and the number of attributes.

13. `-T 1`: It computes the *fraction of maximum covering spheres*. Its computational cost depends on the product of the number of examples power two and the number of attributes.

14. `-T 2`: It computes the *average number of points per dimension*. Its computation cost is constant.

15. `-A`: This option runs all the complexity measures.

16. `-d`: It splits m-class data sets into m two-class data sets and applies the complexity measures to these two-class data sets. If the input data set has two classes, the activation of this option has no effect on the run.

All these measures can be run on data sets that contain different types of attributes. The following subsection explains the types of distance functions that have been included in the library.

Distance Function Options

We extended the code to be able to efficiently deal with continuous and nominal/categorical attributes. For this reason, we implemented several distance functions which have been shown to enhance the behavior of instance-based classifiers (Wilson and Martinez, 1997). Regardless of the type of the attributes, the distance between two examples is calculated as

$$\text{distance}(ex_1, ex_2) = \sqrt{\sum_{a=1}^{\ell} d_a(ex_1^a, ex_2^a)^2}, \tag{13}$$

where ex_1 and ex_2 are the two input examples, ℓ is the number of input attributes, and d_a is the function that computes the distance between two values of the attribute a.

For *continuous attributes*, we implemented the following distance functions (option `-cM` permits specifying the distance function to be used):

1. `-cM 1`: It implements the Euclidean distance function, which is defined as

$$d_a(ex_1^a, ex_2^a) = ex_1^a - ex_2^a, \tag{14}$$

   where ex_i^a refers to the value of the attribute a of the example i. Note that we do not return the absolute value of the substraction.

2. `-cM 2` *(default option)*: It implements the normalized Euclidean distance function (i.e., the distance of each attribute is normalized by its range), which is defined as

$$d_a(ex_1^a, ex_2^a) = \frac{(ex_1^a - ex_2^a)}{range_a}, \tag{15}$$

   where $range_a$ is the range of attribute a.

3. `-cM 3`: It implements the Euclidean distance function weighted by the standard deviation, which is defined as

$$d_a(ex_1^a, ex_2^a) = \frac{(ex_1^a - ex_2^a)}{4\sigma_a}, \tag{16}$$

   where σ_a is the standard deviation of the attribute a. This metric is said to be less sensitive to outliers. Since 95% of values in a normal distribution fall within two standard deviations of the mean, the difference between numeric values is divided by 4 standard deviations to scale each value into a range that is usually of width 1.

For *nominal attributes* (option `-nM`), we implemented the following distance functions:

1. `-nM 1` *(default option)*: It implements the overlap distance function. That is, if two nominal attributes are equal, the distance between them is 0. Otherwise, the distance is 1.

$$d_a(ex_1^a, ex_2^a) = \begin{cases} 0 & \text{if } ex_1^a = ex_2^a, \\ 1 & \text{otherwise.} \end{cases} \tag{17}$$

2. `-nM 2`: It implements the VDM distance function (Stanfill and Waltz, 1986) defined as

$$d_a(ex_1^a, ex_2^a) = \sum_{c=1}^{C} |P(a, ex_1^a, c) - P(a, ex_2^a, c)|, \tag{18}$$

where C is the total number of classes in the problem, and $P(a, ex_i^a, c)$ is the conditional probability that the output class is c given that the attribute a has the value ex_i^a. That is, $P(a, ex_i^a, c)$ is defined as

$$P(a, ex_i^a, c) = \frac{N(a, ex_i^a, c)}{N(a, ex_i^a)}, \tag{19}$$

where $N(a, ex_1^a, c)$ is the number of instances of class c for which the attribute a takes the value ex_i^a, and $N(a, ex_i^a)$ is the total number of instances for which the attribute a takes the value ex_i^a.

The application also permits mapping the nominal values into integers and applying an Euclidean distance (option `-nM 3`) or a normalized Euclidean distance (option `-nM 4`).

Note that by combining the normalized Euclidean distance for continuous attributes with the overlap distance for nominal attributes, we obtain the *Heterogeneous Euclidean-Overlap Metric* (HEOM) (Aha et al., 1991; Giraud-Carrier and Martinez, 1995). Moreover, the combination of the normalized Euclidean distance for continuous attributes with the VDM distance for nominal attributes results in the *Heterogeneous Value Difference Metric* (HVDM) (Wilson and Martinez, 1997).

Other Options

In addition to the aforementioned options, the application also provides other functionalities, which are detailed as follows.

1. `-t2class`: This option requires that the input data set have m classes ($m > 2$). It generates m new data sets (one for each class), which consist of examples of one class against examples of all the other classes. Therefore, it creates two-class data sets in which each concept is discriminated against the rest of the domain. The name of the output file is determined by option `-o`.

2. `-cv numFolds`: This option runs a stratified cross-validation with *numFolds* folds (where *numFolds* has to be greater than 1). The name of the output file is determined by option `-o`. If *numFolds* is not specified or incorrectly set (i.e., *numFolds* < 2), it is set to 10 by default.

3. `-s num`: It changes the random seed to the specified integer *num*. If *num* = 0, the seed is randomly generated using the system time.

4. `-D`: It prints debug information to screen while running the application.

5. `-B`: It permits running the complexity measures for multiple data sets. The file specified with option `-i` is not considered as a data set, but as a file that contains the paths to many data sets (one path in each line). Thus, the application is run with the specified options for each one of the data sets.

6. `-nRU`: It indicates that the missing values do not have to be replaced. If this option is not specified, the library automatically replaces (1) missing values of continuous attributes with the average value of the attribute for the class of the example and (2) missing values of nominal attributes with the median value of the attribute for the class of the example.

3.3.3 Format of the Output

The output depends on the type of run.

- For runs concerning complexity measures, the application writes an output file which specifies the value for each measure. In the next section there are several examples of output files.

- For runs concerning options `-t2class` and `-cv`, different files are created with the corresponding output data sets. The root name of the output file is the one specified with option `-o`.

By default the system saves the computation of the complexity measures in a .txt format file. However, there are two more formats available:

1. `-latex`: It saves the results in a .tex format file (see Figures 2 and 3). By compiling this file with a LaTeX compiler, the resulting document shows the output in a table view.

2. `-xml`: It saves the output in a .xml format file (see Figures 4 and 5).

In any case, information of the process will be displayed on the screen. For more detailed information, option `-D` must be used.

Finally, warnings and errors will be reported in file <`output_file`>`.log` where <`output_file`> is the output file name specified by option `-o`.

3.3.4 Dealing with Missing Values

As some measures do not accept missing values, we apply the following policy to replace them. For real- and integer-valued attributes, a missing value in the attribute a of an instance that predicts class c is replaced with the average value of the attribute a computed on all the instances that predict class c. For nominal attributes, it is replaced with the median value of the attribute a computed on all the instances that predict class c.

```
\documentclass{article}
\usepackage{longtable}

\begin{document}

\begin{center}
\footnotesize{
\setlength\LTleft{-100pt}

\begin{longtable}{lrrrrrrrrrrrrr}

% First header
\caption[Complexity measures]{Summary of the complexity measures computation} \\
\label{tab:CMresults} \\
\hline
\textbf{Data set name} & \textbf{F1} & \textbf{F1v} & \textbf{F2} & \textbf{F3} & \textbf{F4} &
\textbf{L1} & \textbf{L2} & \textbf{L3} & \textbf{N1} & \textbf{N2} & \textbf{N3} & \textbf{N4} &
\textbf{T1} & \textbf{T2} \\
\hline
\hline
\endfirsthead

% Next headers
\multicolumn{15}{c}{{\bfseries \tablename\ \thetable{} -- continued from previous page}} \\
\hline
\textbf{Data set name} & \textbf{F1} & \textbf{F1v} & \textbf{F2} & \textbf{F3} & \textbf{F4} &
\textbf{L1} & \textbf{L2} & \textbf{L3} & \textbf{N1} & \textbf{N2} & \textbf{N3} & \textbf{N4} &
\textbf{T1} & \textbf{T2} \\
\hline
\hline
\endhead

% First footer
\hline
\multicolumn{15}{r}{{Continued on next page}} \\
\hline
\endfoot

% Last footer
\hline
\hline
\endlastfoot

../Data/wbcd.dat              & 3.568 & 0.068 & 0.248 & 0.119 & 0.232 & 0.457 & 0.034 & 0.009 &
0.059 & 0.335 & 0.041 & 0.032 & 0.801 & 77.667 \\
../Data/pim.dat               & 0.576 & 1.414 & 0.252 & 0.007 & 0.022 & 0.689 & 0.350 & 0.500 &
0.438 & 0.840 & 0.294 & 0.289 & 0.999 & 96.000 \\
\hline

\end{longtable}

}
\end{center}

\end{document}
```

Figure 2: Output of the command `./dcol -i ./list.dat -o ./Output/list -B -A -latex`

Table 1: Summary of the complexity measures computation

Data set name	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
../Data/wbcd.dat	3.568	0.068	0.248	0.119	0.232	0.457	0.034	0.009	0.059	0.335	0.041	0.032	0.801	77.667
../Data/pim.dat	0.576	1.414	0.252	0.007	0.022	0.689	0.350	0.500	0.438	0.840	0.294	0.289	0.999	96.000

Figure 3: Final output of the command `./dcol -i ./list.dat -o ./Output/list -B -A -latex`

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ComplexityAnalysis>
<Dataset>
<Name> ../Data/wbcd.dat </Name>
<F1> 3.568 </F1>
<F1v> 0.068 </F1v>
<F2> 0.248 </F2>
<F3> 0.119 </F3>
<F4> 0.232 </F4>
<L1> 0.457 </L1>
<L2> 0.034 </L2>
<L3> 0.009 </L3>
<N1> 0.059 </N1>
<N2> 0.335 </N2>
<N3> 0.041 </N3>
<N4> 0.032 </N4>
<T1> 0.801 </T1>
<T2> 77.667 </T2>
</Dataset>
<Dataset>
<Name> ../Data/pim.dat </Name>
<F1> 0.576 </F1>
<F1v> 1.414 </F1v>
<F2> 0.252 </F2>
<F3> 0.007 </F3>
<F4> 0.022 </F4>
<L1> 0.689 </L1>
<L2> 0.350 </L2>
<L3> 0.500 </L3>
<N1> 0.438 </N1>
<N2> 0.840 </N2>
<N3> 0.294 </N3>
<N4> 0.289 </N4>
<T1> 0.999 </T1>
<T2> 96.000 </T2>
</Dataset>
</ComplexityAnalysis>
```

Figure 4: Output of the command `./dcol -i ./list.dat -o ./Output/list -B -A -xml`

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <ComplexityAnalysis>
  - <Dataset>
      <Name>../Data/wbcd.dat</Name>
      <F1>3.568</F1>
      <F1v>0.068</F1v>
      <F2>0.248</F2>
      <F3>0.119</F3>
      <F4>0.232</F4>
      <L1>0.457</L1>
      <L2>0.034</L2>
      <L3>0.009</L3>
      <N1>0.059</N1>
      <N2>0.335</N2>
      <N3>0.041</N3>
      <N4>0.032</N4>
      <T1>0.801</T1>
      <T2>77.667</T2>
    </Dataset>
  - <Dataset>
      <Name>../Data/pim.dat</Name>
      <F1>0.576</F1>
      <F1v>1.414</F1v>
      <F2>0.252</F2>
      <F3>0.007</F3>
      <F4>0.022</F4>
      <L1>0.689</L1>
      <L2>0.350</L2>
      <L3>0.500</L3>
      <N1>0.438</N1>
      <N2>0.840</N2>
      <N3>0.294</N3>
      <N4>0.289</N4>
      <T1>0.999</T1>
      <T2>96.000</T2>
    </Dataset>
  </ComplexityAnalysis>
```

Figure 5: Final output of the command `./dcol -i ./list.dat -o ./Output/list -B -A -xml`

3.3.5 Dealing with Multiple-Class Data Sets

Originally, all the complexity measures were defined for two-class data sets. In our implementation, we extended all these measures, except for the three that involve the construction of a SVM and the directional-vector maximum Fisher's discriminant ratio, to m-class data sets. Notwithstanding, applying these measures to m-class data sets may hinder some key observations on the complexity related to individual classes. In order to avoid this, we included option `-d`. When this option is specified and a data set with more than two classes is fed to the application, we first transform the m-class data set into m two-class data sets by creating a new two-class data set for each class, which contains all the examples of the class and groups all the remaining examples in a new class. Therefore, each data set focuses on the complexity of a single class, considering that we may map each different learning concept into a class.

3.4 How to Call the Code

In the following, we give some recommendations to call the complexity measure routines and the partitioning and stratified k-fold cross-validation functionalities from own code.

All the files except `InputOptions.h` and `InputOptions.cpp` are required. To run any complexity measure or additional functionality, three steps have to be done:

1. Include `ComplexityMeasures.h` in the code.
2. Load the input data set.
3. Call the methods of the complexity measure or of the other two functionalities.

The programmer has to insert `#include ''ComplexityMeasures.h''` into to the file where the call will be made. Then, the data set has to be load by calling the constructor of the *ComplexityMeasures* class. Finally, the desired method has to be called; some of them require to pass parameters. The detail of the parameters and the returned values of each method is specified in the html/pdf documentation of the library.

Figure 6 shows an example for every complexity measure call.

4 Running Some Examples

Below, we show different scenarios with the generated output. For all the examples, we use three data sets extracted from the UCI repository: the Wisconsin breast-cancer database (*wbcd*), the pima Indian diabetes (*pim*), and the iris problem (*iris*). *wbcd* has 699 instances and 9 integer-valued attributes. *pim* has 768 instances and 8 real-valued attributes. *iris* has 150 instances and 4 real-valued attributes. All these data sets are provided with the library source code.

4.1 Compute Several Measures on a Data Set

To run a group of complexity measures on a single data set, we can type

```
./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -F 2 -N 1
```

In this example, `../Data/wbcd.dat` is the input data set, the result of the complexity measures will be written in `./Output/wbcd.txt`, and the complexity measures F2 and N1 will be computed with the default distance functions for continuous and nominal attributes (i.e., normalized Euclidean

```
int main ( int argc, char** argv ) {

    ComplexityMeasures* dSet;

    float F1, F1v, F2, F3, F4, L1, L2, L3, N1, N2, N3, N4, T1, T2;
    float* vectorResults;
    float  measureResult;
    int    att, i;

    std::string inputDatasetName;
    bool replaceUnknownValues;
    int typeOfContinuousDistFunction;
    int typeOfNominalDistFunction;

    std::string outputDatasetName;
    int foldsCV;

    // Indicate the path of the input data set and specify the options for the complexity measures.
    inputDatasetName = "./test.arff";
    replaceUnknownValues = true;
    typeOfContinuousDistFunction = 2;
    typeOfNominalDistFunction = 1;

    // For the support vector machine, we need the weights (w) and the offset (b).
    float* w = 0;
    float b = 0;

    // Load the specified input data set.
    dSet = new ComplexityMeasures ( inputDatasetName, true, replaceUnknownValues, typeOfContinuousDistFunction, typeOfNominalDistFunction );


    /*** COMPLEXITY MEASURES ***/

    // F1 - Maximum Fisher's discriminant ratio.
    F1 = dSet->computeFisher ( att );

    // F1v - Directional-vector maximum Fisher's discriminant ratio.
    F1v = dSet->computeFisherVectorized();

    // F2 - Overlap of the per-class bounding boxes.
    F2  = dSet->computeVolumeOverlap();

    // F3 - Maximum (individual) feature efficiency.
    vectorResults  = dSet->computeMaximumEfficiencyOfAttributes ( att, measureResult );
    delete [] vectorResults;
    F3 = measureResult;

    // F4 - Collective feature efficiency.
    vectorResults = dSet->computeMaximumEfficiencyOfAttributes ( att, measureResult );
    measureResult = 0;

    for ( i = 0; i < dSet->getNumberOfAttributes(); i++ ) {
        measureResult += vectorResults[i] / dSet->getNumberOfExamples();
    }

    delete [] vectorResults;
    F4 = measureResult;

    // L1 - Minimized sum of the error distance of a linear classifier.
    w  = dSet->trainSMO ( b );
    L1 = dSet->computeNonLinearityLCDistance ( w, b );
    // delete [] w;

    // L2 - Training error of a linear classifier.
    // w  = dSet->trainSMO ( b );
    L2 = dSet->computeNonLinearityLCTrain ( w, b );
    // delete [] w;

    // L3 - Nonlinearity of a linear classifier.
    // w  = dSet->trainSMO ( b );
    L3 = dSet->computeNonLinearityLCConvexHull ( w, b );
    delete [] w;
```

```
    // N1 - Fraction of points on the class boundary.
    int** spanningTree = dSet->computePrim();
    N1 = dSet->computeBoundary ( spanningTree );
    for ( i = 0; i < dSet->getNumberOfExamples() - 1; i++ ) {
        delete [] spanningTree[i];
    }
    delete [] spanningTree;

    // N2 - Ratio of average intra/inter class nearest neighbor distance.
    N2 = dSet->computeIntraInter();

    // N3 - Leave-one-out error rate of the one-nearest neighbor classifier.
    N3 = dSet->computeNonLinearityKNNTrain();

    // N4 - Nonlinearity of the one-nearest neighbor classifier.
    N4 = dSet->computeNonLinearityKNNConvexHull ();

    // T1 - Fraction of maximum covering spheres
    vectorResults = dSet->computeFractMaxCoveringSpheres();
    T1 = vectorResults[0] / dSet->getNumberOfExamples();
    delete [] vectorResults;

    // T2 - Average number of points per dimension.
    T2 = dSet->averageNumberOfSamplesPerDimension();


    /*** OTHER FUNCTIONALITIES ***/

    // Indicate the name of the output data set and specify the options.
    outputDatasetName = "outputTest";
    foldsCV = 10;

    // Partitioning.
    dSet->generate2ClassDatasets ( outputDatasetName );

    // Stratified k-fold cross-validation.
    dSet->stratifiedCrossValidation ( foldsCV, outputDatasetName );
    dSet->deleteExamplesPerClass();

   return 1;

} // end main
```

Figure 6: Recommendations to call DCoL routines from own code.

distance for continuous attributes and overlap distance for nominal attributes). The distance functions can be changed with options `-cM` and `-nM`. For instance, to use a simple Euclidean distance, we should type

```
./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -F 2 -N 1 -cM 1
```

The content written in the file `./Output/wbcd.txt` is shown in Figure 7.

```
TABLE LEGEND
  F2:  Overlap of the per-class bounding boxes
  N1:  Fraction of points on the class boundary


DATA SET                              F2          N1
../Data/wbcd.dat                   0.248       0.062
```

Figure 7: Output of the command `./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -F 2 -N 1 -cM 1`

4.2 Compute All the Measures on a Data Set

To run all the complexity measures on a single data set, we can type

```
./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -A
```

which is equivalent to specify all the complexity measures, i.e.,

```
./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -F 1 -F 1v -F 2 -F 3 -F 4
-L 1 -L 2 -L 3 -N 1 -N 2 -N 3 -N 4 -T 1 -T 2
```

The result of this run is shown in Figure 8.

```
TABLE LEGEND
  F1:  Maximum Fisher's discriminant ratio
  F1v: Directional-vector maximum Fisher's discriminant ratio
  F2:  Overlap of the per-class bounding boxes
  F3:  Maximum (individual) feature efficiency
  F4:  Collective feature efficiency (sum of each feature efficiency)
  L1:  Minimized sum of the error distance of a linear classifier (linear SMO)
  L2:  Training error of a linear classifier (linear SMO)
  L3:  Nonlinearity of a linear classifier (linear SMO)
  N1:  Fraction of points on the class boundary
  N2:  Ratio of average intra/inter class nearest neighbor distance
  N3:  Leave-one-out error rate of the one-nearest neighbor classifier
  N4:  Nonlinearity of the one-nearest neighbor classifier
  T1:  Fraction of maximum covering spheres
  T2:  Average number of points per dimension


DATA SET              F1   F1v    F2    F3    F4    L1    L2    L3    N1    N2    N3    N4    T1     T2
../Data/wbcd.dat   3.568 0.068 0.248 0.119 0.232 0.457 0.034 0.009 0.059 0.335 0.041 0.032 0.801 77.667
```

Figure 8: Output of the command `./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -A`

4.3 Compute the Measures on a Collection of Data Sets

To run the measures on a collection of data sets, we can type

```
./dcol -i ./list.dat -o ./Output/list -B -A
```

where `list.dat` contains a data set (file and path) for each line, and option `-B` indicates that the input file contains a list of data sets instead of the information of a single data set. Figure 9 shows an example of this file. Thus, all the specified measures will be run on each data set. In this example, all the complexity measures will be applied to each data set specified in the input file since option `-A` is used. The output is a table that gathers the value of the complexity measures for each data set (see Figure 10). A group of selected complexity measures can be run if desired. For example,

```
./dcol -i ./list.dat -o ./Output/list -N 1 -N 2 -B
```

runs N1 and N2 on each of the data sets specified in the file `list.dat`.

```
../Data/wbcd.dat
../Data/pim.dat
```

Figure 9: Example of file `list.dat` that contains a list of paths to data sets which the complexity measures will be applied to.

In case of including m-class data sets, it is **strongly recommended** to activate option `-d`, which forces the application to discriminate each class against the others and, consequently, allows

```
TABLE LEGEND
  F1:  Maximum Fisher's discriminant ratio
  F1v: Directional-vector maximum Fisher's discriminant ratio
  F2:  Overlap of the per-class bounding boxes
  F3:  Maximum (individual) feature efficiency
  F4:  Collective feature efficiency (sum of each feature efficiency)
  L1:  Minimized sum of the error distance of a linear classifier (linear SMO)
  L2:  Training error of a linear classifier (linear SMO)
  L3:  Nonlinearity of a linear classifier (linear SMO)
  N1:  Fraction of points on the class boundary
  N2:  Ratio of average intra/inter class nearest neighbor distance
  N3:  Leave-one-out error rate of the one-nearest neighbor classifier
  N4:  Nonlinearity of the one-nearest neighbor classifier
  T1:  Fraction of maximum covering spheres
  T2:  Average number of points per dimension


DATA SET                F1   F1v    F2    F3    F4    L1    L2    L3    N1    N2    N3    N4    T1     T2
../Data/wbcd.dat     3.568 0.068 0.248 0.119 0.232 0.457 0.034 0.009 0.059 0.335 0.041 0.032 0.801 77.667
../Data/pim.dat      0.576 1.414 0.252 0.007 0.022 0.689 0.350 0.500 0.438 0.840 0.294 0.289 0.999 96.000
```

Figure 10: Output of the command `./dcol -i ./list.dat -o ./Output/list -B -A`

us to analyze the contribution of each class individually. To illustrate this, Figure 11 shows a list of data sets that includes the iris problem, which contains 3 classes. Then, if we run the following command:

```
./dcol -i ./list2.dat -o ./Output/list2 -B -A -d
```

we obtain the output depicted in Figure 12. Note that the results for the two-class problems, i.e., *wbcd* and *pim*, are the same as those presented in Figure 10; thence, option `-d` has no effect on two-class data sets. On the other hand, the iris problem has been split into three data sets (*iris.dat.2c0*, *iris.dat.2c1*, and *iris.dat.2c2*) each one concerning one of the classes of the problem.

```
../Data/wbcd.dat
../Data/pim.dat
../Data/iris.dat
```

Figure 11: Example of file `list2.dat` that contains a list of paths to data sets which the complexity measures will be applied to.

4.4 Compute Cross-Validation on a Data Set

To run cross-validation, we should type

```
./dcol -i ../Data/wbcd.dat -o ./Output/wbcd -cv 10
```

that runs a stratified 10-fold cross-validation and generates 20 data sets whose names are `./Output/wbcd-K-Xtra.dat` and `./Output/wbcd-K-Xtst.dat`, where K is the number of folds specified (in the example K=10) and X corresponds to the current fold (its value ranges from 0 to K-1). If the number of folds is not specified, the application sets it to 10 by default. This operation is not compatible with option `-B`.

```
TABLE LEGEND
  F1:  Maximum Fisher's discriminant ratio
  F1v: Directional-vector maximum Fisher's discriminant ratio
  F2:  Overlap of the per-class bounding boxes
  F3:  Maximum (individual) feature efficiency
  F4:  Collective feature efficiency (sum of each feature efficiency)
  L1:  Minimized sum of the error distance of a linear classifier (linear SMO)
  L2:  Training error of a linear classifier (linear SMO)
  L3:  Nonlinearity of a linear classifier (linear SMO)
  N1:  Fraction of points on the class boundary
  N2:  Ratio of average intra/inter class nearest neighbor distance
  N3:  Leave-one-out error rate of the one-nearest neighbor classifier
  N4:  Nonlinearity of the one-nearest neighbor classifier
  T1:  Fraction of maximum covering spheres
  T2:  Average number of points per dimension


DATA SET                 F1    F1v    F2    F3    F4    L1    L2    L3    N1    N2    N3    N4    T1     T2
../Data/wbcd.dat      3.568  0.068 0.248 0.119 0.232 0.457 0.034 0.009 0.059 0.335 0.041 0.032 0.801 77.667
../Data/pim.dat       0.576  1.414 0.252 0.007 0.022 0.689 0.350 0.500 0.438 0.840 0.294 0.289 0.999 96.000
../Data/iris.dat.2c0 16.822 13.385 0.005 0.573 0.573 0.643 0.333 0.500 0.013 0.076 0.000 0.000 0.313 37.500
../Data/iris.dat.2c1  0.677  0.324 0.035 0.560 0.887 0.667 0.333 0.500 0.107 0.174 0.073 0.177 0.913 37.500
../Data/iris.dat.2c2  3.935  6.836 0.007  0.75 0.913 0.662 0.333 0.500 0.093 0.136 0.073 0.030 0.893 37.500
```

Figure 12: Output of the command `./dcol -i ./list2.dat -o ./Output/list2 -B -A -d`

4.5 Create Two-Class Data Sets from a Multiple-Class Data Set

Finally, we show an example of how to create m two-class data sets from an m-class data set. To achieve this, we should type

```
./dcol -i ../Data/iris.dat -o ./Output/iris -t2class
```

that generates m data sets whose names are `./Output/iris.2cx`, where x goes from 0 to $m-1$. This operation is not compatible with option `-B`.

5 The Structure of the Code

This section summarizes the object oriented design of the code. It is worth highlighting that the application does not use third party libraries for compatibility across different architectures. Figure 13 provides the class diagram of the application. In what follows, we explain the role of each class. For the sake of clarity, we divide the classes into the following three groups: (1) application core classes, (2) distance functions classes, and (3) auxiliary classes.

Application core classes.

1. **Dataset** contains all the information of the data set itself and some basic functions to deal with the data.
2. **ExtendedDataset** extends the *Dataset* class by implementing methods (1) to organize the examples per class, (2) to print the information to output files or to the screen, and (3) to get the distance between two examples.
3. **ComplexityMeasures** extends the *ExtendedDataset* class by adding all the complexity measures.

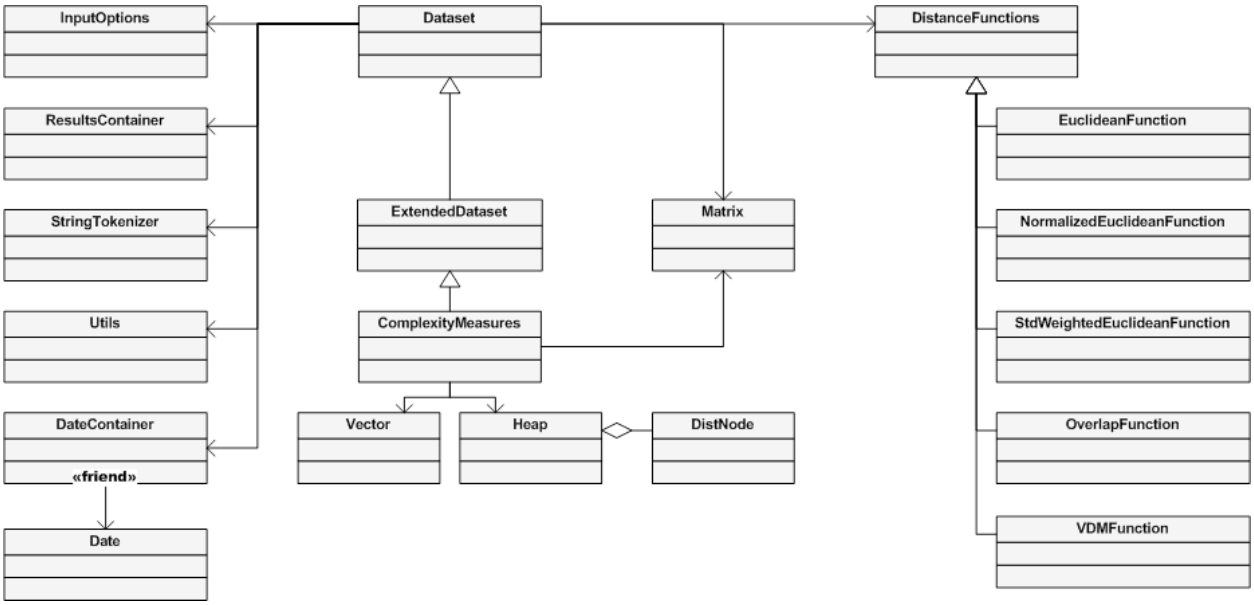

Figure 13: Class diagram of the library.

Distance functions classes.

4. **DistanceFunctions** is an abstract class that defines the methods required to implement a new distance function. We provide five implementations of this abstract class that correspond to the five distance functions proposed in Section 3.3.2, i.e., *EuclideanFunction*, *NormalizedEuclideanFunction*, *StdWeightedEuclideanFunction*, *OverlapFunction*, and *VDMFunction*. These classes are placed in the `DistanceFunctions` folder.

Auxiliary classes.

5. **DateContainer** reads dates from strings formatted in a prefixed way.
6. **Date** contains the information of a date.
7. **DistNode** is used as the node of the max heap.
8. **Heap** implements a max heap. It is used by the kNN classifier.
9. **InputOptions** parses the command line to get the options specified by the user.
10. **Matrix** represents a matrix of continuous numbers and implements several methods such as operations between matrices or inverse and pseudoinverse computation.
11. **ResultsContainer** permits keeping the results obtained from the application of the complexity measures.
12. **StringTokenizer** implements a string tokenizer, that is to say, it splits a string line into tokens separated by the given delimiter.
13. **Utils** implements a series of utilities.
14. **Vector** implements a dynamic vector of elements.

A brief description of the main methods of each class is given below. For further information, please see the html or pdf documentation provided in the `Help` folder.

5.1 Dataset Class

The *Dataset* class contains all the routines to (1) read, store, and manipulate the data and (2) make statistics (e.g., average, median, and standard deviation values per attribute and per class). The different methods were implemented in two separate 'cpp' files: (1) `Dataset.cpp` and (2) `Statistics.cpp`. The main methods coded in each of the two files are enumerated in the following two subsections.

5.1.1 Dataset.cpp File

This file implements routines to load data sets, provide some information about the stored data, and calculate preliminary statistics. The main methods are:

1. *readData*: It reads the data set in KEEL or WEKA format (except for string attributes) from a specified file.
2. *getNumberOfExamples*: It returns the number of examples of the data set.
3. *getNumberOfAttributes*: It returns the number of attributes of the data set.
4. *getNumberOfClasses*: It returns the number of classes of the data set.
5. *makeInitialStatistics*: It computes a set of statistics on the data (e.g., the average and the standard deviation of continuous attributes and the median and the frequency of each value for nominal attributes, among others). This information is used to compute the complexity measures.

5.1.2 Statistics.cpp File

This file implements routines to extract statistics from the data. The main methods are *calculateAverages* and *calculateDeviations*, which calculate the average and standard deviation for each attribute and class.

5.2 ExtendedDataset Class

This class extends the *Dataset* class and provides new methods (1) to organize the examples per class and apply partition procedures, (2) to print the information to output files or to the screen, and (3) to get the distance between two examples. The main methods are:

1. *print*: It prints the data set to a file.
2. *getDistance:* It returns the distance between two examples. The distance function specified by the user is employed to compute the distance.
3. *getApproximateDistance:* It returns the approximate distance between two examples. This is used to speed up the process in complexity measures that call the distance routine for a large number of pairs of examples.
4. *stratifiedCrossValidation*: It runs a stratified k-fold cross-validation and prints all the training and test folds.
5. *printOneClassAgainstOthers*: Given an m-class data set ($m > 2$), it generates m two-class data sets and prints them to different files.

5.3 ComplexityMeasures Class

The *ComplexityMeasures* inherits from the *ExtendedDataset* class, providing the implementation of all the complexity measures. The methods are implemented in two different 'cpp' files: (1) `ComplexityMeasures.cpp` and (2) `SMO.cpp`. The main methods coded in each of the two files are enumerated in the following two subsections.

5.3.1 ComplexityMeasures.cpp File

The most important methods related to the complexity measures are:

1. *computeFisher*: It computes the maximum Fisher's discriminant ratio among the different attributes (F1).
2. *computeFisherVectorized*: It computes the maximum Fisher's discriminant ratio projected over the vector with the best direction (F1v).
3. *computeVolumeOverlap*: It computes the overlap of the per-class bounding boxes (F2).
4. *computeMaximumEfficiencyOfAttributes*: It computes the maximum (individual) feature efficiency (F3) and the collective feature efficiency (F4).
5. *computeNonLinearityLCDistance:* It computes the sum of the error distance of a linear classifier (L1).
6. *computeNonLinearityLCTrain:* It computes the training error of a linear classifier (L2).
7. *computeNonLinearityLCConvexHull:* It computes the nonlinearity of a linear classifier (L3).
8. *computeBoundary*: It computes the fraction of points on the class boundary (N1).
9. *computeIntraInter*: It computes the ratio of the average intra/inter class nearest neighbor distance (N2).
10. *computeNonLinearityKNNTrain*: It computes the leave-one-out error rate of the one-nearest neighbor classifier (N3).
11. *computeNonLinearityKNNConvexHull*: It computes the nonlinearity of the one-nearest neighbor classifier (N4).
12. *computePretopology*: It computes the fraction of maximum covering spheres (T1).
13. *averageNumberOfSamplesPerDimension*: It computes the average number of examples per dimension (T2).

5.3.2 SMO.cpp File

This file contains the implementation for the *Sequential Minimal Optimization* algorithm (Platt, 1998) to train support vector machines. The main routine of this file is *trainSMO*, which implements the main loop of the SVM training.

5.4 DateContainer Class

This class implements methods to transform date strings into numerics or *Date* objects. The main methods are:

1. *transformToNumeric*: It transforms a date into a numeric (float).
2. *transformToLongNumeric*: It transforms a date into a numeric (long double).
3. *transformToDate*: It transforms a date into a *Date* object.

5.5 Date Class

This class implements a node for the *DateContainer* class. It stores the information of a date. The main methods are:

1. *operator* <: It returns whether the first node is greater than the second.
2. *operator* >: It returns whether the first node is less than the second.
3. *operator* <=: It returns whether the first node is greater than or equal to the second.
4. *operator* >=: It returns whether the first node is less than or equal to the second.
5. *operator* ==: It returns whether the two nodes are equal.
6. *operator* !=: It returns whether the two nodes are different.

5.6 Distance Function Classes

We provide five implementations of the *DistanceFunction* abstract class that correspond to the five distance functions proposed in Section 3.3.2, i.e., *EuclideanFunction*, *NormalizedEuclideanFunction*, *StdWeightedEuclideanFunction*, *OverlapFunction*, and *VDMFunction*. These functions are placed in the `DistanceFunctions` folder. All them calculate the distance between two attributes.

5.7 DistNode Class

This class implements a node for the Heap class. It stores the index and the distance between two examples. The main methods are:

1. *operator* <: It returns whether the first node is greater than the second.
2. *operator* >: It returns whether the first node is less than the second.
3. *operator* ==: It returns whether the two nodes are equal.

5.8 Heap Class

This class implements a max heap. The methods for this class are implemented in the file `Heap.cpp`. The main methods are:

1. *add*: It adds one element at the end of the heap and shifts it up.
2. *remove*: It removes the first element of the max heap, i.e., the biggest element of the heap.

3. *getNumberOfElements*: It returns the number of elements stored in the heap.
4. *getFirst*: It returns the first element of the heap.

5.9 InputOptions Class

The main method of this class is *parseInput*, which parses the command line.

5.10 Matrix Class

This class implements an $m \times n$ matrix. The main methods are:

1. *computeInverse*: It computes the inverse of the matrix.
2. *computeGaussJordan*: It computes the Gauss-Jordan elimination on the matrix.
3. *computePseudoInverse*: It computes the pseudoinverse of the matrix.
4. *calculateSVD*: It computes the singular value decomposition of the matrix.
5. *operator* $+$: It computes the sum of two matrices.
6. *operator* $-$: It computes the substraction of two matrices.
7. *operator* $*$: It computes the product of two matrices.

5.11 ResultsContainer Class

This class stores the results of runs over a single or multiple data sets. The main methods are:

1. *addElement*: It adds a new result.
2. *getResult*: It returns a particular result.

5.12 StringTokenizer Class

This class implements a string tokenizer. The main methods are:

1. *getNextToken*: It returns the next token.
2. *countTokens*: It returns the total number of tokens.
3. *hasMoreTokens*: It returns whether there are more tokens or not.

5.13 Utils Class

This class implements some utilities. The main methods are:

1. *quickSort*: It implements the *quicksort* sorting algorithm.
2. *max*: It returns the maximal element between those passed as parameters.
3. *min*: It returns the minimal element between those passed as parameters.
4. *f_rand*: It generates a real-valued random number between 0 and 1.
5. *i_rand*: It generates an integer-valued random number.
6. *trim*: It removes spaces at the beginning and at the end of a string.

5.14 Vector Class

This class implements a dynamic vector. The main methods are:

1. *addElement*: It adds one element to the vector.
2. *removeElement*: It removes one element from the vector.
3. *size*: It returns the number of elements in the vector.

6 How to Extend the Code

We encourage the users to extend the code by incorporating new complexity measures, new distance functions, and other functionalities. As follows, we briefly explain how the code should be extended to provide the software with new functionalities.

New complexity measures should be implemented in the *ComplexityMeasures* class. This class inherits from *ExtendedDataset*, which in turn extends *Dataset*. Therefore, in the *ComplexityMeasures* class there are several functions that calculate averages and standard deviations per attribute and per class that may be useful for the implementation of new measures. In addition to the inclusion of new measures in the *ComplexityMeasures* class, also the command line options, which are managed by the class *InputOptions*, should be extended to be able to consider the new complexity measures. First, the function *parseInput* of the class *InputOptions* should be modified to understand the new options, and these should be stored as an attribute of the class. Moreover, functions that indicate whether the corresponding measure has been selected should be coded. There are some guidelines in the code—find `[NEW COMPLEXITY MEASURE]`—that can help to update the basics for a new complexity measure preserving the same coding style and nomenclature.

New distance functions should inherit the abstract class *DistanceFunction*, define a constructor and a destructor, and implement the function *computeDistance*, which computes the distance between two attributes. The function *instantiateDistanceFunctions* of the class *Dataset* should be modified to accept the new implemented distance functions. Besides, the class *InputOptions* must be modified as explained above to accept the new command line options. Note that the function *distance* in `Dataset.cpp` computes the distance between two examples by returning the square root of the sum of the distance between each pair of attributes power two. This function should not be modified although new distance functions are introduced into the code.

Finally, new functionalities can be implemented within the *Dataset* or the *ExtendedDataset* classes, or new classes containing a *Dataset* object can be implemented.

7 Copyright

```
% This is an example of a data set in KEEL format

@relation paint
@attribute colour {yellow, white, black}
@attribute amount integer [1, 10]
@attribute density real [0.1, 2.5]
@outputs colour
@data
yellow, 4, 1.2
black, 1, 2.1
yellow, 2, 0.8
white, 8, 0.9
white, ?, 0.9
```

Figure 14: Example of a data set in KEEL format.

A KEEL Format

The KEEL format permits defining data sets for classification and regression. This format proposes to structure data sets in two parts: (1) the data set definition and (2) the data itself.

Data set definition. It is composed of the data set name, the definition of all the attributes that describe the problem, and, optionally, the definition of the goal of each attribute.

- Name of the data set: **@relation** <name_ds>
- Definition of each attribute: attributes can be continuous, integer, or nominal.
  - **@attribute** <name_att> **real** *[min, max]*
  - **@attribute** <name_att> **integer** *[min, max]*
  - **@attribute** <name_att> **{**v_1, v_2, ..., v_n**}**

  where <name_att> is the name of the attribute (the name cannot contain commas). For real- and integer-valued attributes, [min, max] **is optional** and defines the minimum and maximum value of the attribute. For nominal attributes, a list of all possible values, i.e., $\{v_1, v_2, ..., v_n\}$, follows the name of the attribute.
- Role of each attribute [optional]: the format permits defining whether the attributes are input or output attributes, by using the **@outputs** tag:
  - **@outputs** <name_att>

  where <name_att> is the name of the output attribute and has to be previously defined as explained above. Since the library is applied to classification tasks, the output attribute has to be either integer-valued or nominal. If the attribute is continuous, an error is generated. If the **@outputs** tag is not specified, the library considers that the last attribute in the definition list is the class attribute.

Data itself. After the definition section, we write the **@data** tag, and then, an instance is written in each line, i.e.:

@data
x_{11}, x_{12}, ... , x_{1N}
x_{21}, x_{22}, ... , x_{2N}

where missing values are represented by symbols *?*, *<null>*, or *<NULL>*. Figure 14 provides an example of a data set in KEEL format.

B Test Report

This appendix reports the results of a large experimentation carried out to show the performance of the library. First, we describe the collection of data sets used in the experimentation, then explain the steps to replicate such experimentation, and finally, illustrate the results obtained from the experiments.

B.1 Test Bed

The collection of data sets used in the experimentation consists of 70 binary real-world problems from the UCI machine learning repository. The problems of m-classes ($m > 2$) were transformed into m two-class problems by discriminating one class against the others. The collection contains a diverse type of problems whose training set size varies from 9 to 48842 instances and whose feature dimensionality varies from 2 to 927 attributes.

The folder `Test` contains a folder named `Datasets` which includes all the data sets of this collection. Table 2 briefly summarizes the following characteristics of the data sets:

- Number of instances (#Inst)
- Total number of attributes (#Att)
- Number of real attributes (#Real)
- Number of integer attributes (#Int)
- Number of nominal attributes (#Nom)
- Percentage of missing attributes (%missAtt)
- Percentage of missing instances (%missInst)
- Percentage of missing values (%missVal)
- Percentage of instances belonging to the majority class (%Maj)
- Percentage of instances belonging to the minority class (%Min)

B.2 Procedure and Parametrization

The experimentation performed consisted in testing all the complexity measures over a wide collection of data sets by varying the distance function parametrization. More specifically, the bash script `measures.sh` runs the following configurations:

```
./dcol -i ./testList.dat -o ./Output/cM1nM1 -cM 1 -nM 1 -A -B -latex
./dcol -i ./testList.dat -o ./Output/cM1nM2 -cM 1 -nM 2 -A -B -latex
./dcol -i ./testList.dat -o ./Output/cM2nM1 -cM 2 -nM 1 -A -B -latex
./dcol -i ./testList.dat -o ./Output/cM2nM2 -cM 2 -nM 2 -A -B -latex
./dcol -i ./testList.dat -o ./Output/cM3nM1 -cM 3 -nM 1 -A -B -latex
./dcol -i ./testList.dat -o ./Output/cM3nM2 -cM 3 -nM 2 -A -B -latex
```

The experiments were run in batch mode, therefore, the file `testList.dat` contains the paths of all the data sets of the collection.

To replicate the experimentation, the user should proceed as follows.

1. Save the bash script `measures.sh` and the file `testList.dat` in the `Source` folder of the DCoL software.
2. Start the run by typing: `> bash measures.sh`

Table 2: Description of the external characteristics of the data set collection.

Dataset	#Inst	#Att	#Real	#Int	#Nom	%missAtt	%mistInst	%missVal	%Maj	%Min
aba.2c0.dat	4177	8	7	0	1	0.00	0.00	0.00	99.98	0.02
adl.dat	48842	14	0	6	8	21.43	7.41	0.95	76.07	23.93
ann.2c0.dat	898	38	6	0	32	0.00	0.00	0.00	99.11	0.89
asbestos.dat	83	3	0	1	2	0.00	0.00	0.00	55.42	44.58
aud.2c0.dat	226	69	0	0	69	10.14	98.23	2.03	78.76	21.24
aut.2c0.dat	205	25	15	0	10	28.00	22.44	1.15	98.54	1.46
authors.2c0.dat	841	70	0	70	0	0.00	0.00	0.00	62.31	37.69
bal.2c0.dat	625	4	4	0	0	0.00	0.00	0.00	53.92	46.08
bankruptcy.dat	50	5	5	0	0	0.00	0.00	0.00	50.00	50.00
benford.2c0.dat	9	6	0	0	6	0.00	0.00	0.00	88.89	11.11
bondrate.2c0.dat	57	10	0	4	6	10.00	1.75	0.18	89.47	10.53
bpa.dat	345	6	6	0	0	0.00	0.00	0.00	57.97	42.03
briv1.2c0.dat	105	11	0	4	7	63.64	33.33	5.28	84.76	15.24
briv2.2c0.dat	105	11	0	1	10	63.64	33.33	5.28	84.76	15.24
car.2c0.dat	1728	6	0	0	6	0.00	0.00	0.00	70.02	29.98
cmc.2c0.dat	1473	9	0	2	7	0.00	0.00	0.00	57.30	42.70
col.dat	368	22	7	0	15	95.45	98.10	23.80	63.04	36.96
crx.dat	690	15	3	3	9	53.33	5.51	0.66	55.51	44.49
drm.2c0.dat	366	34	0	33	1	2.94	2.19	0.06	69.40	30.60
ech.dat	74	11	6	2	3	72.73	17.57	2.83	67.57	32.43
euc.2c0.dat	736	19	7	7	5	47.37	12.91	3.20	75.54	24.46
flg.2c0.dat	194	28	0	10	18	0.00	0.00	0.00	79.38	20.62
fourclass.dat	862	2	2	0	0	0.00	0.00	0.00	64.39	35.61
gls.2c0.dat	214	9	9	0	0	0.00	0.00	0.00	67.29	32.71
gru.2c0.dat	155	8	0	2	6	0.00	0.00	0.00	68.39	31.61
h-s.dat	270	13	13	0	0	0.00	0.00	0.00	55.56	44.44
hab.dat	306	3	0	3	0	0.00	0.00	0.00	73.53	26.47
hay.2c0.dat	159	4	0	0	4	0.00	0.00	0.00	59.75	40.25
hep.dat	155	19	2	4	13	78.95	48.39	5.67	79.35	20.65
hrs.dat	368	27	8	4	15	77.78	98.10	19.39	63.04	36.96
ion.dat	351	34	34	0	0	0.00	0.00	0.00	64.10	35.90
irs.2c0.dat	150	4	4	0	0	0.00	0.00	0.00	66.67	33.33
krk.2c0.dat	28056	6	0	0	6	0.00	0.00	0.00	90.03	9.97
krkp.dat	3196	36	0	0	36	0.00	0.00	0.00	52.22	47.78
liv.dat	345	6	1	5	0	0.00	0.00	0.00	57.97	42.03
lng.2c0.dat	32	56	0	0	56	3.57	15.63	0.28	71.88	28.13
mag.dat	19020	10	10	0	0	0.00	0.00	0.00	64.84	35.16
msh.dat	8124	22	0	0	22	4.55	30.53	1.39	51.80	48.20
nrs.2c0.dat	12960	8	0	0	8	0.00	0.00	0.00	66.67	33.33
opt.2c0.dat	5620	64	0	64	0	0.00	0.00	0.00	90.14	9.86
pas.2c0.dat	36	22	15	6	1	0.00	0.00	0.00	66.67	33.33
pbc.2c0.dat	5473	10	4	6	0	0.00	0.00	0.00	89.77	10.23
pen.2c0.dat	10992	16	0	16	0	0.00	0.00	0.00	89.60	10.40
pim.dat	768	8	8	0	0	0.00	0.00	0.00	65.10	34.90
pos.2c0.dat	90	8	0	1	7	12.50	3.33	0.42	97.78	2.22
seg.2c0.dat	2310	19	19	0	0	0.00	0.00	0.00	85.71	14.29
shs.2c0.dat	52	24	6	15	3	29.17	3.85	0.56	55.77	44.23
shu.2c0.dat	52	23	5	15	3	34.78	17.31	3.26	53.85	46.15
spa.dat	4601	57	55	2	0	0.00	0.00	0.00	60.60	39.40
spect.dat	267	22	0	0	22	0.00	0.00	0.00	79.40	20.60
spectf.dat	267	44	0	44	0	0.00	0.00	0.00	79.40	20.60
statlog-sgm.2c0.dat	2310	19	19	0	0	0.00	0.00	0.00	85.71	14.29
tae.2c0.dat	151	5	0	1	4	0.00	0.00	0.00	67.55	32.45
tao.dat	1888	2	2	0	0	0.00	0.00	0.00	50.00	50.00
thy.2c0.dat	215	5	5	0	0	0.00	0.00	0.00	69.77	30.23
tis.dat	13375	927	924	0	3	0.00	0.00	0.00	75.24	24.76
tnc.dat	2201	3	0	0	3	0.00	0.00	0.00	67.70	32.30
trn.dat	10	32	0	10	22	31.25	70.00	15.94	50.00	50.00
veh.2c0.dat	846	18	18	0	0	0.00	0.00	0.00	74.94	25.06
vot.dat	435	16	0	0	16	100.00	46.67	5.63	61.38	38.62
wav21.2c0.dat	5000	21	21	0	0	0.00	0.00	0.00	66.86	33.14
wav40.2c0.dat	5000	40	40	0	0	0.00	0.00	0.00	66.16	33.84
wbcd.dat	699	9	0	9	0	11.11	2.29	0.25	65.52	34.48
wdbc.dat	569	30	30	0	0	0.00	0.00	0.00	62.74	37.26
whi.2c0.dat	63	31	27	0	4	0.00	0.00	0.00	60.32	39.68
win.2c0.dat	178	13	11	2	0	0.00	0.00	0.00	66.85	33.15
wne.2c0.dat	178	13	13	0	0	0.00	0.00	0.00	66.85	33.15
wpbc.dat	198	33	33	0	0	3.03	2.02	0.06	76.26	23.74
yea.2c0.dat	1484	8	8	0	0	0.00	0.00	0.00	68.80	31.20
zoo.2c0.dat	101	16	0	0	16	0.00	0.00	0.00	59.41	40.59

B.3 Results

This section contains the results of the DCoL run over the collection of data sets introduced in Subsection B.1 and after following the procedure and system parametrization specified in Subsection B.2.

Table 3: Complexity measures results contained in the file `./Output/cM1nM1.tex`
Table 4: Complexity measures results contained in the file `./Output/cM1nM2.tex`
Table 5: Complexity measures results contained in the file `./Output/cM2nM1.tex`
Table 6: Complexity measures results contained in the file `./Output/cM2nM2.tex`
Table 7: Complexity measures results contained in the file `./Output/cM3nM1.tex`
Table 8: Complexity measures results contained in the file `./Output/cM3nM2.tex`

Acknowledgements

The authors would like to thank the support of *Ministerio de Ciencia y Tecnología* under projects TIN2005-08386-C05-04 and TIN2008-06681-C06-05, *Generalitat de Catalunya* under grant 2005SGR-00302, *Fundació Crèdit Andorrà*, and *Govern d'Andorra*.

References

D. W. Aha, D. F. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 13(3):307–318, 2008. ISSN 1432-7643.

A. Asuncion and D. Newman. UCI machine learning repository [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 2007.

M. Basu and T. K. Ho, editors. *Data complexity in pattern recognition*. Springer, 2006.

J. Casillas, A. Orriols-Puig, and E. Bernadó-Mansilla. Toward evolving consistent, complete, and compact fuzzy rule sets for classification problems. In *3rd International Workshop on Genetic and Evolving Fuzzy Systems (GEFS'08)*, pages 89–94, 2008.

L. Devroye. Automatic pattern recognition: A study of the probability of error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):530–543, 1988.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

C. Giraud-Carrier and T. Martinez. An efficient metric for heterogeneous inductive learning applications in the attribute-value language. In *Intelligent Systems (Proceedings of GWIC'94)*, pages 341–350. Kluwer Academic Publishers, 1995.

T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

Table 3: `./dcol -i ./testList.dat -o ./Output/cM1nM1 -cM 1 -nM 1 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	38.853	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.027	2.394e-04	0.000	0.712	522.125
adl.dat	0.346	5.438	0.233	0.030	0.035	1.065	0.166	0.275	0.394	0.173	0.256	0.407	0.994	3488.714
ann.2c0.dat	2.526	54.719	0.000	0.728	1.000	0.099	0.009	0.500	0.013	0.007	0.004	0.200	0.812	23.632
asbestos.dat	0.316	2.029	0.885	0.012	0.012	0.581	0.193	0.229	0.482	0.486	0.253	0.307	0.843	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.235	0.607	0.119	0.175	1.000	3.275
aut.2c0.dat	2.061	18.047	0.000	0.985	1.000	0.034	0.015	0.500	0.044	0.084	0.029	0.444	0.546	8.200
authors.2c0.dat	1.877	40.160	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.019	0.675	0.005	0.017	1.000	12.014
bal.2c0.dat	0.385	6.759	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	4.344	0.001	0.620	0.960	0.922	0.160	0.220	0.300	0.531	0.140	0.080	0.860	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.556	0.831	0.333	0.444	1.000	1.500
bondrate.2c0.dat	0.904	3.911	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.316	0.399	0.228	0.518	0.526	5.700
bpa.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.568	0.926	0.377	0.329	1.000	57.500
briv1.2c0.dat	3.234	42.100	0.000	0.714	0.714	0.377	0.019	0.014	0.143	0.084	0.114	0.176	0.657	9.545
briv2.2c0.dat	0.415	37.043	0.000	0.714	0.714	0.395	0.010	0.024	0.200	0.433	0.086	0.129	0.990	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.234	0.902	0.175	0.472	1.000	288.000
cmc.2c0.dat	0.029	0.637	0.750	0.002	0.002	0.788	0.356	0.383	0.527	0.874	0.383	0.380	0.998	163.667
col.dat	0.305	9.086	0.187	0.038	0.098	0.561	0.073	0.030	0.514	0.733	0.340	0.330	0.997	16.727
crx.dat	0.364	3.711	0.001	0.032	0.068	0.290	0.145	0.154	0.546	0.182	0.335	0.331	0.987	46.000
drm.2c0.dat	7.789	64.965	0.000	0.541	0.989	0.323	0.000	0.000	0.041	0.531	0.011	0.004	0.995	10.765
ech.dat	4.221	18.386	0.000	0.973	1.000	0.362	0.054	0.014	0.122	0.310	0.068	0.014	0.743	6.727
euc.2c0.dat	2.722	8.526	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.220	0.479	0.115	0.269	0.985	38.737
flg.2c0.dat	1.545	2.682	0.000	0.093	0.232	0.444	0.206	0.500	0.423	1.028	0.278	0.448	0.990	6.929
fourclass.dat	0.952	2.038	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.227	0.472	431.000
gls.2c0.dat	0.649	1.682	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.318	0.379	0.196	0.273	0.995	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.477	0.893	0.329	0.319	1.000	19.375
hab.dat	0.185	0.503	0.718	0.029	0.033	0.530	0.265	0.500	0.497	0.795	0.320	0.359	0.922	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.384	0.442	0.258	0.450	1.000	39.750
hep.dat	1.040	7.056	0.000	0.232	0.568	0.467	0.194	0.490	0.303	0.727	0.213	0.223	0.968	8.158
hrs.dat	0.305	36.950	0.000	0.038	0.141	0.546	0.073	0.037	0.383	0.482	0.226	0.329	0.916	13.630
h-s.dat	0.760	4.819	0.196	0.015	0.093	0.532	0.156	0.104	0.604	0.899	0.426	0.337	0.996	20.769
ion.dat	0.614	7.040	0.000	0.191	0.994	0.453	0.117	0.150	0.234	0.635	0.134	0.174	0.946	10.324
irs.2c0.dat	16.822	43.567	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.079	0.000	0.000	0.527	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.545	0.846	0.462	0.487	1.000	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.273	0.714	0.156	0.264	1.000	88.778
liv.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.568	0.926	0.377	0.333	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.469	0.962	0.375	0.000	1.000	0.571
mag.dat	0.559	2.162	0.081	0.006	0.018	0.717	0.214	0.233	0.333	0.635	0.215	0.255	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.377	0.000	0.033	1.000	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	1.000	7.716e-05	0.000	1.000	1620.000
opt.2c0.dat	4.729	37.658	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.408	1.779e-04	0.003	0.991	87.812
pas.2c0.dat	1.628	10.856	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.333	0.375	0.250	0.139	0.972	1.636
pbc.2c0.dat	0.511	9.753	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.066	0.212	0.038	0.229	0.995	547.300
pen.2c0.dat	3.475	12.146	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.163	3.639e-04	0.019	0.899	687.000
pim.dat	0.576	1.911	0.252	0.007	0.022	0.689	0.350	0.500	0.467	0.828	0.320	0.317	1.000	96.000
pos.2c0.dat	0.172	2.418	0.000	0.478	0.733	0.045	0.022	0.500	0.044	0.348	0.022	0.500	1.000	11.250
seg.2c0.dat	1.798	11.943	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.025	0.106	0.005	0.079	0.957	121.579
shs.2c0.dat	1.127	9.692	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.558	0.853	0.385	0.317	1.000	2.167
shu.2c0.dat	6.788	6.315	0.005	0.250	0.885	0.748	0.404	0.423	0.481	0.890	0.327	0.163	1.000	2.261
spa.dat	0.347	5.557	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.308	0.343	0.170	0.271	0.975	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.337	0.785	0.270	0.476	1.000	12.136
spectf.dat	0.553	2.197	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.337	0.803	0.251	0.099	1.000	6.068
statlog-sgm.2c0.dat	1.798	12.532	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.025	0.106	0.005	0.082	0.957	121.579
tae.2c0.dat	6.316	0.608	0.559	0.079	0.146	0.653	0.325	0.500	0.464	0.465	0.225	0.454	0.993	30.200
tao.dat	1.395	4.916	0.479	0.360	0.362	0.590	0.163	0.110	0.091	0.157	0.045	0.155	0.398	944.000
thy.2c0.dat	0.256	3.434	0.001	0.186	0.414	0.588	0.302	0.500	0.140	0.318	0.051	0.177	0.898	43.000
tis.dat	0.472	7.253	0.000	0.033	0.234	1.415	0.070	0.084	0.324	0.826	0.214	0.023	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.033	0.677	0.500	1.000	733.667
trn.dat	18.000	8.269	0.000	0.500	0.500	0.706	0.000	0.000	0.500	0.900	0.400	0.000	1.000	0.312
veh.2c0.dat	0.185	2.214	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.396	0.643	0.281	0.395	1.000	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.115	0.348	0.064	0.008	1.000	27.188
wav21.2c0.dat	1.182	3.493	0.036	0.123	0.183	1.020	0.141	0.086	0.231	0.786	0.166	0.107	1.000	238.095
wav40.2c0.dat	1.168	3.681	0.015	0.149	0.211	1.000	0.143	0.081	0.235	0.877	0.168	0.066	1.000	125.000
wbcd.dat	3.568	23.285	0.248	0.119	0.232	0.457	0.034	0.012	0.062	0.335	0.043	0.030	0.801	77.667
wdbc.dat	3.405	19.512	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.135	0.160	0.084	0.105	0.794	18.967
whi.2c0.dat	2.188	10.441	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.540	0.920	0.333	0.087	1.000	2.032
win.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.135	0.079	0.084	0.039	0.618	13.692
wne.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.135	0.079	0.084	0.059	0.618	13.692
wpbc.dat	0.472	3.320	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.485	0.785	0.308	0.391	0.995	6.000
yea.2c0.dat	0.214	0.743	0.056	0.127	0.177	0.624	0.312	0.500	0.453	0.716	0.300	0.336	1.000	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.200	0.000	0.000	1.000	6.312

Table 4: `./dcol -i ./testList.dat -o ./Output/cM1nM2 -cM 1 -nM 2 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	38.853	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.026	2.394e-04	0.000	0.683	522.125
adl.dat	0.346	5.438	0.233	0.030	0.035	1.065	0.166	0.275	0.394	0.173	0.256	0.407	1.000	3488.714
ann.2c0.dat	2.526	54.719	0.000	0.728	1.000	0.099	0.009	0.500	0.013	0.007	0.004	0.201	0.878	23.632
asbestos.dat	0.316	2.029	0.885	0.012	0.012	0.581	0.193	0.229	0.494	0.483	0.253	0.313	1.000	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.208	0.499	0.102	0.447	0.996	3.275
aut.2c0.dat	2.061	18.047	0.000	0.985	1.000	0.034	0.015	0.500	0.044	0.084	0.029	0.444	0.546	8.200
authors.2c0.dat	1.877	40.160	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.019	0.675	0.005	0.017	1.000	12.014
bal.2c0.dat	0.385	6.759	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	4.344	0.001	0.620	0.960	0.922	0.160	0.220	0.300	0.531	0.140	0.080	0.860	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.222	0.000	0.111	0.000	1.000	1.500
bondrate.2c0.dat	0.904	3.911	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.316	0.399	0.228	0.518	0.526	5.700
bpa.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.568	0.926	0.377	0.329	1.000	57.500
briv1.2c0.dat	3.234	42.100	0.000	0.714	0.714	0.377	0.019	0.014	0.143	0.084	0.114	0.181	0.581	9.545
briv2.2c0.dat	0.415	37.043	0.000	0.714	0.714	0.395	0.010	0.024	0.133	0.354	0.076	0.119	0.990	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.041	0.016	0.009	0.150	0.968	288.000
cmc.2c0.dat	0.029	0.637	0.750	0.002	0.002	0.788	0.356	0.383	0.541	0.785	0.371	0.390	0.997	163.667
col.dat	0.305	9.086	0.187	0.038	0.098	0.561	0.073	0.030	0.514	0.725	0.342	0.332	1.000	16.727
crx.dat	0.364	3.711	0.001	0.032	0.068	0.290	0.145	0.154	0.541	0.182	0.329	0.325	0.988	46.000
drm.2c0.dat	7.789	64.965	0.000	0.541	0.989	0.323	0.000	0.000	0.041	0.533	0.011	0.004	0.995	10.765
ech.dat	4.221	18.386	0.000	0.973	1.000	0.362	0.054	0.014	0.122	0.309	0.068	0.014	0.824	6.727
euc.2c0.dat	2.722	8.526	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.220	0.479	0.117	0.269	0.982	38.737
flg.2c0.dat	1.545	2.682	0.000	0.093	0.232	0.444	0.206	0.500	0.423	1.028	0.289	0.459	0.990	6.929
fourclass.dat	0.952	2.038	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.227	0.472	431.000
gls.2c0.dat	0.649	1.682	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.318	0.379	0.196	0.273	0.995	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.387	0.519	0.252	0.355	1.000	19.375
hab.dat	0.185	0.503	0.718	0.029	0.033	0.530	0.265	0.500	0.497	0.795	0.320	0.359	0.922	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.352	0.361	0.277	0.270	1.000	39.750
hep.dat	1.040	7.056	0.000	0.232	0.568	0.467	0.194	0.490	0.303	0.727	0.213	0.223	0.968	8.158
hrs.dat	0.305	36.950	0.000	0.038	0.141	0.546	0.073	0.037	0.383	0.482	0.226	0.329	0.918	13.630
h-s.dat	0.760	4.819	0.196	0.015	0.093	0.532	0.156	0.104	0.604	0.899	0.426	0.337	0.996	20.769
ion.dat	0.614	7.040	0.000	0.191	0.994	0.453	0.117	0.150	0.234	0.635	0.134	0.174	0.946	10.324
irs.2c0.dat	16.822	43.567	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.079	0.000	0.000	0.527	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.214	0.233	0.131	0.279	0.937	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.168	0.442	0.074	0.290	1.000	88.778
liv.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.568	0.926	0.377	0.333	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.375	0.923	0.250	0.000	1.000	0.571
mag.dat	0.559	2.162	0.081	0.006	0.018	0.717	0.214	0.233	0.333	0.635	0.215	0.255	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.018	0.000	0.064	0.840	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	0.000	0.000	0.000	1.000	1620.000
opt.2c0.dat	4.729	37.658	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.408	1.779e-04	0.003	0.991	87.812
pas.2c0.dat	1.628	10.856	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.333	0.375	0.250	0.139	0.972	1.636
pbc.2c0.dat	0.511	9.753	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.066	0.212	0.038	0.229	0.995	547.300
pen.2c0.dat	3.475	12.146	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.163	3.639e-04	0.019	0.899	687.000
pim.dat	0.576	1.911	0.252	0.007	0.022	0.689	0.350	0.500	0.467	0.828	0.320	0.317	1.000	96.000
pos.2c0.dat	0.172	2.418	0.000	0.478	0.733	0.045	0.022	0.500	0.067	0.171	0.056	0.506	1.000	11.250
seg.2c0.dat	1.798	11.943	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.025	0.106	0.005	0.079	0.957	121.579
shs.2c0.dat	1.127	9.692	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.558	0.853	0.385	0.317	1.000	2.167
shu.2c0.dat	6.788	6.315	0.005	0.250	0.885	0.748	0.404	0.423	0.481	0.890	0.327	0.163	1.000	2.261
spa.dat	0.347	5.557	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.308	0.343	0.170	0.271	0.975	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.348	0.773	0.270	0.551	1.000	12.136
spectf.dat	0.553	2.197	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.337	0.803	0.251	0.099	1.000	6.068
statlog-sgm.2c0.dat	1.798	12.532	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.025	0.106	0.005	0.082	0.957	121.579
tae.2c0.dat	6.316	0.608	0.559	0.079	0.146	0.653	0.325	0.500	0.437	0.445	0.192	0.427	0.974	30.200
tao.dat	1.395	4.916	0.479	0.360	0.362	0.590	0.163	0.110	0.091	0.157	0.045	0.155	0.398	944.000
thy.2c0.dat	0.256	3.434	0.001	0.186	0.414	0.588	0.302	0.500	0.140	0.318	0.051	0.177	0.898	43.000
tis.dat	0.472	7.253	0.000	0.033	0.234	1.415	0.070	0.084	0.326	0.825	0.218	0.024	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.015	0.677	0.500	1.000	733.667
trn.dat	18.000	8.269	0.000	0.500	0.500	0.706	0.000	0.000	0.200	0.651	0.000	0.000	1.000	0.312
veh.2c0.dat	0.185	2.214	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.396	0.643	0.281	0.395	1.000	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.071	0.249	0.051	0.056	1.000	27.188
wav21.2c0.dat	1.182	3.493	0.036	0.123	0.183	1.020	0.141	0.086	0.231	0.786	0.166	0.107	1.000	238.095
wav40.2c0.dat	1.168	3.681	0.015	0.149	0.211	1.000	0.143	0.081	0.235	0.877	0.168	0.066	1.000	125.000
wbcd.dat	3.568	23.285	0.248	0.119	0.232	0.457	0.034	0.012	0.062	0.335	0.043	0.030	0.801	77.667
wdbc.dat	3.405	19.512	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.135	0.160	0.084	0.105	0.794	18.967
whi.2c0.dat	2.188	10.441	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.540	0.920	0.333	0.087	1.000	2.032
win.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.135	0.079	0.084	0.039	0.618	13.692
wne.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.135	0.079	0.084	0.059	0.618	13.692
wpbc.dat	0.472	3.320	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.485	0.785	0.308	0.391	0.995	6.000
yea.2c0.dat	0.214	0.743	0.056	0.127	0.177	0.624	0.312	0.500	0.453	0.716	0.300	0.336	1.000	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.108	0.000	0.000	0.970	6.312

Table 5: `./dcol -i ./testList.dat -o ./Output/cM2nM1 -cM 2 -nM 1 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	33.605	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.026	2.394e-04	0.000	0.648	522.125
adl.dat	0.346	2.725	0.233	0.030	0.035	1.065	0.166	0.275	0.290	0.446	0.201	0.269	0.996	3488.714
ann.2c0.dat	2.526	59.494	0.000	0.728	1.000	0.099	0.009	0.500	0.007	0.135	0.001	0.021	0.960	23.632
asbestos.dat	0.316	1.549	0.885	0.012	0.012	0.581	0.193	0.229	0.398	0.562	0.229	0.199	0.602	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.235	0.607	0.119	0.175	1.000	3.275
aut.2c0.dat	2.061	14.036	0.000	0.985	1.000	0.034	0.015	0.500	0.049	0.285	0.020	0.010	1.000	8.200
authors.2c0.dat	1.877	19.663	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.012	0.772	0.004	5.945e-04	1.000	12.014
bal.2c0.dat	0.385	0.418	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	1.673	0.001	0.620	0.960	0.922	0.160	0.220	0.240	0.632	0.140	0.090	0.920	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.556	0.831	0.333	0.444	1.000	1.500
bondrate.2c0.dat	0.904	1.386	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.228	0.813	0.158	0.167	1.000	5.700
bpa.dat	0.055	0.148	0.073	0.032	0.107	0.841	0.420	0.500	0.574	0.913	0.374	0.342	1.000	57.500
briv1.2c0.dat	3.234	57.275	0.000	0.714	0.714	0.377	0.019	0.014	0.095	0.264	0.048	0.005	1.000	9.545
briv2.2c0.dat	0.415	44.742	0.000	0.714	0.714	0.395	0.010	0.024	0.076	0.379	0.048	0.014	1.000	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.234	0.902	0.175	0.472	1.000	288.000
cmc.2c0.dat	0.029	0.289	0.750	0.002	0.002	0.788	0.356	0.383	0.583	0.877	0.406	0.370	0.944	163.667
col.dat	0.305	12.965	0.187	0.038	0.098	0.561	0.073	0.030	0.201	0.689	0.114	0.030	1.000	16.727
crx.dat	0.364	8.962	0.001	0.032	0.068	0.290	0.145	0.154	0.290	0.563	0.181	0.117	0.999	46.000
drm.2c0.dat	7.789	3.031	0.000	0.541	0.989	0.323	0.000	0.000	0.016	0.443	0.005	0.000	0.981	10.765
ech.dat	4.221	15.976	0.000	0.973	1.000	0.362	0.054	0.014	0.135	0.433	0.095	0.000	0.973	6.727
euc.2c0.dat	2.722	16.211	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.192	0.336	0.113	0.158	0.999	38.737
flg.2c0.dat	1.545	4.211	0.000	0.093	0.232	0.444	0.206	0.500	0.294	0.770	0.175	0.157	1.000	6.929
fourclass.dat	0.952	1.372e-04	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.229	0.452	431.000
gls.2c0.dat	0.649	0.579	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.322	0.432	0.182	0.227	0.991	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.497	0.825	0.342	0.300	0.994	19.375
hab.dat	0.185	0.002	0.718	0.029	0.033	0.530	0.265	0.500	0.539	0.754	0.353	0.368	0.931	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.384	0.442	0.258	0.450	1.000	39.750
hep.dat	1.040	6.249	0.000	0.232	0.568	0.467	0.194	0.490	0.284	0.623	0.187	0.026	0.987	8.158
hrs.dat	0.305	17.615	0.000	0.038	0.141	0.546	0.073	0.037	0.193	0.684	0.106	0.024	1.000	13.630
h-s.dat	0.760	4.344	0.196	0.015	0.093	0.532	0.156	0.104	0.367	0.672	0.244	0.107	1.000	20.769
ion.dat	0.614	3.728	0.000	0.191	0.994	0.453	0.117	0.150	0.231	0.631	0.131	0.162	0.946	10.324
irs.2c0.dat	16.822	25.477	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.096	0.000	0.000	0.453	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.545	0.846	0.462	0.487	1.000	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.273	0.714	0.156	0.264	1.000	88.778
liv.dat	0.055	0.148	0.073	0.032	0.107	0.841	0.420	0.500	0.574	0.913	0.374	0.322	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.469	0.962	0.375	0.000	1.000	0.571
mag.dat	0.559	58.946	0.081	0.006	0.018	0.717	0.214	0.233	0.293	0.650	0.188	0.236	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.377	0.000	0.033	1.000	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	1.000	7.716e-05	0.000	1.000	1620.000
opt.2c0.dat	4.729	0.050	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.408	1.779e-04	0.003	0.991	87.812
pas.2c0.dat	1.628	15.391	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.306	0.646	0.194	0.028	1.000	1.636
pbc.2c0.dat	0.511	176.800	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.068	0.184	0.037	0.213	0.988	547.300
pen.2c0.dat	3.475	0.001	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.163	3.639e-04	0.019	0.899	687.000
pim.dat	0.576	1.414	0.252	0.007	0.022	0.689	0.350	0.500	0.438	0.840	0.294	0.289	0.999	96.000
pos.2c0.dat	0.172	2.138	0.000	0.478	0.733	0.045	0.022	0.500	0.056	0.415	0.033	0.500	1.000	11.250
seg.2c0.dat	1.798	18.828	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.114	0.003	0.038	0.933	121.579
shs.2c0.dat	1.127	-3.500e+00	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.327	0.789	0.154	0.183	1.000	2.167
shu.2c0.dat	6.788	9.850	0.005	0.250	0.885	0.748	0.404	0.423	0.519	0.899	0.288	0.125	1.000	2.261
spa.dat	0.347	12.078	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.155	0.376	0.082	0.113	0.986	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.337	0.785	0.270	0.476	1.000	12.136
spectf.dat	0.553	6.159e-04	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.337	0.803	0.251	0.099	1.000	6.068
statlog-sgm.2c0.dat	1.798	11.204	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.114	0.003	0.037	0.933	121.579
tae.2c0.dat	6.316	0.814	0.559	0.079	0.146	0.653	0.325	0.500	0.450	0.378	0.199	0.427	0.960	30.200
tao.dat	1.395	0.031	0.479	0.360	0.362	0.590	0.163	0.110	0.077	0.157	0.043	0.155	0.398	944.000
thy.2c0.dat	0.256	0.543	0.001	0.186	0.414	0.588	0.302	0.500	0.102	0.310	0.028	0.151	0.837	43.000
tis.dat	0.472	7.410	0.000	0.033	0.234	1.415	0.070	0.084	0.345	0.853	0.260	0.013	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.033	0.677	0.500	1.000	733.667
trn.dat	18.000	4.734	0.000	0.500	0.500	0.706	0.000	0.000	0.400	0.832	0.100	0.000	1.000	0.312
veh.2c0.dat	0.185	1.006	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.365	0.712	0.248	0.353	0.999	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.115	0.348	0.064	0.008	1.000	27.188
wav21.2c0.dat	1.182	0.150	0.036	0.123	0.183	1.020	0.141	0.086	0.238	0.795	0.170	0.109	1.000	238.095
wav40.2c0.dat	1.168	0.130	0.015	0.149	0.211	1.000	0.143	0.081	0.273	0.900	0.196	0.062	1.000	125.000
wbcd.dat	3.568	0.068	0.248	0.119	0.232	0.457	0.034	0.012	0.059	0.335	0.041	0.030	0.801	77.667
wdbc.dat	3.405	56.726	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.072	0.558	0.047	0.022	0.998	18.967
whi.2c0.dat	2.188	6.654	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.492	0.948	0.333	0.063	1.000	2.032
win.2c0.dat	4.290	22.879	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.067	0.490	0.028	0.003	0.994	13.692
wne.2c0.dat	4.290	22.879	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.067	0.490	0.028	0.022	0.994	13.692
wpbc.dat	0.472	4.823	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.424	0.914	0.278	0.217	1.000	6.000
yea.2c0.dat	0.214	0.736	0.056	0.127	0.177	0.624	0.312	0.500	0.450	0.710	0.296	0.340	1.000	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.200	0.000	0.000	1.000	6.312

Table 6: `./dcol -i ./testList.dat -o ./Output/cM2nM2 -cM 2 -nM 2 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	33.605	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.025	2.394e-04	0.000	0.610	522.125
adl.dat	0.346	2.725	0.233	0.030	0.035	1.065	0.166	0.275	0.291	0.467	0.199	0.297	0.997	3488.714
ann.2c0.dat	2.526	59.494	0.000	0.728	1.000	0.099	0.009	0.500	0.007	0.095	0.004	0.096	0.970	23.632
asbestos.dat	0.316	1.549	0.885	0.012	0.012	0.581	0.193	0.229	0.398	0.481	0.229	0.175	0.663	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.208	0.499	0.102	0.447	0.996	3.275
aut.2c0.dat	2.061	14.036	0.000	0.985	1.000	0.034	0.015	0.500	0.029	0.168	0.010	0.000	1.000	8.200
authors.2c0.dat	1.877	19.663	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.012	0.772	0.004	5.945e-04	1.000	12.014
bal.2c0.dat	0.385	0.418	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	1.673	0.001	0.620	0.960	0.922	0.160	0.220	0.240	0.632	0.140	0.090	0.920	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.222	0.000	0.111	0.000	1.000	1.500
bondrate.2c0.dat	0.904	1.386	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.175	0.581	0.123	0.289	0.982	5.700
bpa.dat	0.055	0.148	0.073	0.032	0.107	0.841	0.420	0.500	0.574	0.913	0.374	0.342	1.000	57.500
briv1.2c0.dat	3.234	57.275	0.000	0.714	0.714	0.377	0.019	0.014	0.038	0.169	0.010	0.000	0.971	9.545
briv2.2c0.dat	0.415	44.742	0.000	0.714	0.714	0.395	0.010	0.024	0.057	0.239	0.038	0.000	0.981	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.041	0.016	0.009	0.150	0.968	288.000
cmc.2c0.dat	0.029	0.289	0.750	0.002	0.002	0.788	0.356	0.383	0.585	0.869	0.403	0.393	0.963	163.667
col.dat	0.305	12.965	0.187	0.038	0.098	0.561	0.073	0.030	0.174	0.583	0.101	0.077	0.997	16.727
crx.dat	0.364	8.962	0.001	0.032	0.068	0.290	0.145	0.154	0.278	0.565	0.186	0.183	0.999	46.000
drm.2c0.dat	7.789	3.031	0.000	0.541	0.989	0.323	0.000	0.000	0.022	0.443	0.005	0.000	0.981	10.765
ech.dat	4.221	15.976	0.000	0.973	1.000	0.362	0.054	0.014	0.135	0.308	0.095	0.014	0.986	6.727
euc.2c0.dat	2.722	16.211	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.129	0.405	0.083	0.201	1.000	38.737
flg.2c0.dat	1.545	4.211	0.000	0.093	0.232	0.444	0.206	0.500	0.289	0.685	0.175	0.291	1.000	6.929
fourclass.dat	0.952	1.372e-04	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.229	0.452	431.000
gls.2c0.dat	0.649	0.579	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.322	0.432	0.182	0.227	0.991	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.419	0.641	0.258	0.381	1.000	19.375
hab.dat	0.185	0.002	0.718	0.029	0.033	0.530	0.265	0.500	0.539	0.754	0.353	0.368	0.931	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.352	0.361	0.277	0.270	1.000	39.750
hep.dat	1.040	6.249	0.000	0.232	0.568	0.467	0.194	0.490	0.329	0.615	0.226	0.158	0.935	8.158
hrs.dat	0.305	17.615	0.000	0.038	0.141	0.546	0.073	0.037	0.177	0.612	0.090	0.077	1.000	13.630
h-s.dat	0.760	4.344	0.196	0.015	0.093	0.532	0.156	0.104	0.367	0.672	0.244	0.107	1.000	20.769
ion.dat	0.614	3.728	0.000	0.191	0.994	0.453	0.117	0.150	0.231	0.631	0.131	0.162	0.946	10.324
irs.2c0.dat	16.822	25.477	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.096	0.000	0.000	0.453	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.214	0.233	0.131	0.279	0.937	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.168	0.442	0.074	0.290	1.000	88.778
liv.dat	0.055	0.148	0.073	0.032	0.107	0.841	0.420	0.500	0.574	0.913	0.374	0.322	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.375	0.923	0.250	0.000	1.000	0.571
mag.dat	0.559	58.946	0.081	0.006	0.018	0.717	0.214	0.233	0.293	0.650	0.188	0.236	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.018	0.000	0.064	0.840	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	0.000	0.000	0.000	1.000	1620.000
opt.2c0.dat	4.729	0.050	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.408	1.779e-04	0.003	0.991	87.812
pas.2c0.dat	1.628	15.391	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.278	0.699	0.167	0.083	1.000	1.636
pbc.2c0.dat	0.511	176.800	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.068	0.184	0.037	0.213	0.988	547.300
pen.2c0.dat	3.475	0.001	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.163	3.639e-04	0.019	0.899	687.000
pim.dat	0.576	1.414	0.252	0.007	0.022	0.689	0.350	0.500	0.438	0.840	0.294	0.289	0.999	96.000
pos.2c0.dat	0.172	2.138	0.000	0.478	0.733	0.045	0.022	0.500	0.056	0.241	0.044	0.500	0.989	11.250
seg.2c0.dat	1.798	18.828	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.114	0.003	0.038	0.933	121.579
shs.2c0.dat	1.127	-3.500e+00	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.346	0.652	0.173	0.240	1.000	2.167
shu.2c0.dat	6.788	9.850	0.005	0.250	0.885	0.748	0.404	0.423	0.404	0.785	0.250	0.125	1.000	2.261
spa.dat	0.347	12.078	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.155	0.376	0.082	0.113	0.986	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.348	0.773	0.270	0.551	1.000	12.136
spectf.dat	0.553	6.159e-04	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.337	0.803	0.251	0.099	1.000	6.068
statlog-sgm.2c0.dat	1.798	11.204	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.114	0.003	0.037	0.933	121.579
tae.2c0.dat	6.316	0.814	0.559	0.079	0.146	0.653	0.325	0.500	0.411	0.218	0.199	0.351	0.967	30.200
tao.dat	1.395	0.031	0.479	0.360	0.362	0.590	0.163	0.110	0.077	0.157	0.043	0.155	0.398	944.000
thy.2c0.dat	0.256	0.543	0.001	0.186	0.414	0.588	0.302	0.500	0.102	0.310	0.028	0.151	0.837	43.000
tis.dat	0.472	7.410	0.000	0.033	0.234	1.415	0.070	0.084	0.351	0.850	0.260	0.040	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.015	0.677	0.500	1.000	733.667
trn.dat	18.000	4.734	0.000	0.500	0.500	0.706	0.000	0.000	0.200	0.512	0.000	0.000	0.800	0.312
veh.2c0.dat	0.185	1.006	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.365	0.712	0.248	0.353	0.999	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.071	0.249	0.051	0.056	1.000	27.188
wav21.2c0.dat	1.182	0.150	0.036	0.123	0.183	1.020	0.141	0.086	0.238	0.795	0.170	0.109	1.000	238.095
wav40.2c0.dat	1.168	0.130	0.015	0.149	0.211	1.000	0.143	0.081	0.273	0.900	0.196	0.062	1.000	125.000
wbcd.dat	3.568	0.068	0.248	0.119	0.232	0.457	0.034	0.012	0.059	0.335	0.041	0.030	0.801	77.667
wdbc.dat	3.405	56.726	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.072	0.558	0.047	0.022	0.998	18.967
whi.2c0.dat	2.188	6.654	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.460	0.890	0.286	0.111	1.000	2.032
win.2c0.dat	4.290	22.879	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.067	0.490	0.028	0.003	0.994	13.692
wne.2c0.dat	4.290	22.879	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.067	0.490	0.028	0.022	0.994	13.692
wpbc.dat	0.472	4.823	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.424	0.914	0.278	0.217	1.000	6.000
yea.2c0.dat	0.214	0.736	0.056	0.127	0.177	0.624	0.312	0.500	0.450	0.710	0.296	0.340	1.000	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.108	0.000	0.000	0.970	6.312

Table 7: `./dcol -i ./testList.dat -o ./Output/cM3nM1 -cM 3 -nM 1 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	38.853	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.037	2.394e-04	0.000	0.479	522.125
adl.dat	0.346	5.438	0.233	0.030	0.035	1.065	0.166	0.275	0.286	0.500	0.197	0.269	0.998	3488.714
ann.2c0.dat	2.526	54.719	0.000	0.728	1.000	0.099	0.009	0.500	0.008	0.124	0.001	0.024	0.949	23.632
asbestos.dat	0.316	2.029	0.885	0.012	0.012	0.581	0.193	0.229	0.398	0.594	0.229	0.199	0.639	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.235	0.607	0.119	0.175	1.000	3.275
aut.2c0.dat	2.061	18.047	0.000	0.985	1.000	0.034	0.015	0.500	0.049	0.293	0.020	0.012	0.995	8.200
authors.2c0.dat	1.877	40.160	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.012	0.787	0.005	0.002	1.000	12.014
bal.2c0.dat	0.385	6.759	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	4.344	0.001	0.620	0.960	0.922	0.160	0.220	0.240	0.627	0.120	0.070	0.920	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.556	0.831	0.333	0.444	1.000	1.500
bondrate.2c0.dat	0.904	3.911	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.228	0.826	0.158	0.149	1.000	5.700
bpa.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.562	0.910	0.368	0.328	1.000	57.500
briv1.2c0.dat	3.234	42.100	0.000	0.714	0.714	0.377	0.019	0.014	0.095	0.273	0.048	0.005	1.000	9.545
briv2.2c0.dat	0.415	37.043	0.000	0.714	0.714	0.395	0.010	0.024	0.076	0.378	0.048	0.014	1.000	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.234	0.902	0.175	0.472	1.000	288.000
cmc.2c0.dat	0.029	0.637	0.750	0.002	0.002	0.788	0.356	0.383	0.568	0.874	0.399	0.365	0.940	163.667
col.dat	0.305	9.086	0.187	0.038	0.098	0.561	0.073	0.030	0.209	0.696	0.117	0.027	1.000	16.727
crx.dat	0.364	3.711	0.001	0.032	0.068	0.290	0.145	0.154	0.290	0.650	0.187	0.109	1.000	46.000
drm.2c0.dat	7.789	64.965	0.000	0.541	0.989	0.323	0.000	0.000	0.025	0.477	0.005	0.000	1.000	10.765
ech.dat	4.221	18.386	0.000	0.973	1.000	0.362	0.054	0.014	0.149	0.484	0.095	0.007	0.959	6.727
euc.2c0.dat	2.722	8.526	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.189	0.422	0.113	0.131	0.999	38.737
flg.2c0.dat	1.545	2.682	0.000	0.093	0.232	0.444	0.206	0.500	0.294	0.782	0.175	0.155	1.000	6.929
fourclass.dat	0.952	2.038	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.227	0.466	431.000
gls.2c0.dat	0.649	1.682	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.299	0.507	0.182	0.241	0.986	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.490	0.829	0.323	0.303	1.000	19.375
hab.dat	0.185	0.503	0.718	0.029	0.033	0.530	0.265	0.500	0.516	0.793	0.356	0.363	0.964	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.384	0.442	0.258	0.450	1.000	39.750
hep.dat	1.040	7.056	0.000	0.232	0.568	0.467	0.194	0.490	0.290	0.646	0.181	0.029	0.987	8.158
hrs.dat	0.305	36.950	0.000	0.038	0.141	0.546	0.073	0.037	0.193	0.691	0.111	0.026	0.997	13.630
h-s.dat	0.760	4.819	0.196	0.015	0.093	0.532	0.156	0.104	0.333	0.782	0.233	0.104	0.996	20.769
ion.dat	0.614	7.040	0.000	0.191	0.994	0.453	0.117	0.150	0.239	0.636	0.134	0.157	0.957	10.324
irs.2c0.dat	16.822	43.567	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.109	0.000	0.000	0.460	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.545	0.846	0.462	0.487	1.000	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.273	0.714	0.156	0.264	1.000	88.778
liv.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.562	0.910	0.368	0.316	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.469	0.962	0.375	0.000	1.000	0.571
mag.dat	0.559	2.162	0.081	0.006	0.018	0.717	0.214	0.233	0.284	0.643	0.179	0.232	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.377	0.000	0.033	1.000	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	1.000	7.716e-05	0.000	1.000	1620.000
opt.2c0.dat	4.729	37.658	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.437	1.779e-04	0.004	0.992	87.812
pas.2c0.dat	1.628	10.856	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.306	0.668	0.222	0.028	1.000	1.636
pbc.2c0.dat	0.511	9.753	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.055	0.198	0.031	0.168	0.986	547.300
pen.2c0.dat	3.475	12.146	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.171	6.368e-04	0.019	0.929	687.000
pim.dat	0.576	1.911	0.252	0.007	0.022	0.689	0.350	0.500	0.449	0.846	0.293	0.282	0.999	96.000
pos.2c0.dat	0.172	2.418	0.000	0.478	0.733	0.045	0.022	0.500	0.056	0.443	0.033	0.500	1.000	11.250
seg.2c0.dat	1.798	11.943	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.147	0.003	0.033	0.952	121.579
shs.2c0.dat	1.127	9.692	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.346	0.785	0.154	0.173	1.000	2.167
shu.2c0.dat	6.788	6.315	0.005	0.250	0.885	0.748	0.404	0.423	0.519	0.896	0.308	0.096	1.000	2.261
spa.dat	0.347	5.557	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.160	0.426	0.082	0.103	0.966	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.337	0.785	0.270	0.476	1.000	12.136
spectf.dat	0.553	2.197	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.393	0.845	0.292	0.090	1.000	6.068
statlog-sgm.2c0.dat	1.798	12.532	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.147	0.003	0.033	0.952	121.579
tae.2c0.dat	6.316	0.608	0.559	0.079	0.146	0.653	0.325	0.500	0.450	0.385	0.205	0.427	0.960	30.200
tao.dat	1.395	4.916	0.479	0.360	0.362	0.590	0.163	0.110	0.091	0.157	0.045	0.155	0.397	944.000
thy.2c0.dat	0.256	3.434	0.001	0.186	0.414	0.588	0.302	0.500	0.102	0.321	0.037	0.160	0.912	43.000
tis.dat	0.472	7.253	0.000	0.033	0.234	1.415	0.070	0.084	0.358	0.872	0.269	0.043	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.033	0.677	0.500	1.000	733.667
trn.dat	18.000	8.269	0.000	0.500	0.500	0.706	0.000	0.000	0.400	0.836	0.100	0.000	1.000	0.312
veh.2c0.dat	0.185	2.214	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.359	0.686	0.251	0.335	1.000	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.115	0.348	0.064	0.008	1.000	27.188
wav21.2c0.dat	1.182	3.493	0.036	0.123	0.183	1.020	0.141	0.086	0.248	0.804	0.177	0.112	1.000	238.095
wav40.2c0.dat	1.168	3.681	0.015	0.149	0.211	1.000	0.143	0.081	0.290	0.913	0.208	0.064	1.000	125.000
wbcd.dat	3.568	23.285	0.248	0.119	0.232	0.457	0.034	0.012	0.064	0.358	0.049	0.026	0.861	77.667
wdbc.dat	3.405	19.512	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.083	0.589	0.049	0.021	0.995	18.967
whi.2c0.dat	2.188	10.441	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.444	0.945	0.349	0.079	1.000	2.032
win.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.073	0.507	0.028	0.006	1.000	13.692
wne.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.073	0.507	0.028	0.017	1.000	13.692
wpbc.dat	0.472	3.320	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.434	0.922	0.278	0.242	1.000	6.000
yea.2c0.dat	0.214	0.743	0.056	0.127	0.177	0.624	0.312	0.500	0.450	0.707	0.288	0.341	0.999	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.200	0.000	0.000	1.000	6.312

Table 8: `./dcol -i ./testList.dat -o ./Output/cM3nM2 -cM 3 -nM 2 -A -B -latex`

Data set	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
aba.2c0.dat	14.032	38.853	0.000	1.000	1.000	0.002	2.394e-04	0.500	4.788e-04	0.036	2.394e-04	0.000	0.457	522.125
adl.dat	0.346	5.438	0.233	0.030	0.035	1.065	0.166	0.275	0.285	0.492	0.193	0.277	0.999	3488.714
ann.2c0.dat	2.526	54.719	0.000	0.728	1.000	0.099	0.009	0.500	0.008	0.090	0.004	0.077	0.961	23.632
asbestos.dat	0.316	2.029	0.885	0.012	0.012	0.581	0.193	0.229	0.398	0.489	0.229	0.175	0.687	27.667
aud.2c0.dat	0.022	16.957	0.000	0.407	0.872	0.462	0.212	0.500	0.208	0.499	0.102	0.447	0.996	3.275
aut.2c0.dat	2.061	18.047	0.000	0.985	1.000	0.034	0.015	0.500	0.029	0.191	0.010	0.000	1.000	8.200
authors.2c0.dat	1.877	40.160	3.858e-13	0.225	0.998	0.413	0.002	5.945e-04	0.012	0.787	0.005	0.002	1.000	12.014
bal.2c0.dat	0.385	6.759	1.000	0.000	0.000	0.570	0.048	0.066	0.184	0.623	0.138	0.101	0.902	156.250
bankruptcy.dat	1.121	4.344	0.001	0.620	0.960	0.922	0.160	0.220	0.240	0.627	0.120	0.070	0.920	10.000
benford.2c0.dat	73.143	9.130	0.000	1.000	1.000	0.236	0.111	0.500	0.222	0.000	0.111	0.000	1.000	1.500
bondrate.2c0.dat	0.904	3.911	9.068e-05	0.474	1.000	0.213	0.105	0.500	0.193	0.604	0.140	0.289	1.000	5.700
bpa.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.562	0.910	0.368	0.328	1.000	57.500
briv1.2c0.dat	3.234	42.100	0.000	0.714	0.714	0.377	0.019	0.014	0.038	0.177	0.010	0.000	0.943	9.545
briv2.2c0.dat	0.415	37.043	0.000	0.714	0.714	0.395	0.010	0.024	0.057	0.237	0.038	0.000	0.990	9.545
car.2c0.dat	0.024	1.308	0.250	0.333	0.556	0.857	0.136	0.153	0.041	0.016	0.009	0.150	0.968	288.000
cmc.2c0.dat	0.029	0.637	0.750	0.002	0.002	0.788	0.356	0.383	0.566	0.853	0.401	0.379	0.961	163.667
col.dat	0.305	9.086	0.187	0.038	0.098	0.561	0.073	0.030	0.182	0.613	0.106	0.077	0.995	16.727
crx.dat	0.364	3.711	0.001	0.032	0.068	0.290	0.145	0.154	0.286	0.632	0.201	0.170	1.000	46.000
drm.2c0.dat	7.789	64.965	0.000	0.541	0.989	0.323	0.000	0.000	0.030	0.477	0.005	0.000	1.000	10.765
ech.dat	4.221	18.386	0.000	0.973	1.000	0.362	0.054	0.014	0.149	0.345	0.095	0.027	1.000	6.727
euc.2c0.dat	2.722	8.526	3.990e-07	0.170	0.471	0.559	0.132	0.217	0.133	0.481	0.092	0.160	0.988	38.737
flg.2c0.dat	1.545	2.682	0.000	0.093	0.232	0.444	0.206	0.500	0.299	0.726	0.186	0.271	1.000	6.929
fourclass.dat	0.952	2.038	0.649	0.239	0.278	0.720	0.215	0.314	0.009	0.091	0.000	0.227	0.466	431.000
gls.2c0.dat	0.649	1.682	3.648e-05	0.290	0.486	0.671	0.327	0.500	0.299	0.507	0.182	0.241	0.986	23.778
gru.2c0.dat	3.240	1.388	0.486	0.077	0.123	0.653	0.316	0.500	0.400	0.639	0.252	0.384	1.000	19.375
hab.dat	0.185	0.503	0.718	0.029	0.033	0.530	0.265	0.500	0.516	0.793	0.356	0.363	0.964	102.000
hay.2c0.dat	0.025	0.686	0.296	0.082	0.195	0.805	0.403	0.500	0.352	0.361	0.277	0.270	1.000	39.750
hep.dat	1.040	7.056	0.000	0.232	0.568	0.467	0.194	0.490	0.310	0.629	0.200	0.123	0.981	8.158
hrs.dat	0.305	36.950	0.000	0.038	0.141	0.546	0.073	0.037	0.201	0.635	0.101	0.073	1.000	13.630
h-s.dat	0.760	4.819	0.196	0.015	0.093	0.532	0.156	0.104	0.333	0.782	0.233	0.104	0.996	20.769
ion.dat	0.614	7.040	0.000	0.191	0.994	0.453	0.117	0.150	0.239	0.636	0.134	0.157	0.957	10.324
irs.2c0.dat	16.822	43.567	0.005	0.573	0.573	0.310	0.000	0.000	0.013	0.109	0.000	0.000	0.460	37.500
krk.2c0.dat	0.011	0.053	1.000	0.000	0.000	0.200	0.100	0.500	0.214	0.233	0.131	0.279	0.937	4676.000
krkp.dat	0.002	6.688	0.000	0.183	0.433	0.753	0.062	0.084	0.168	0.442	0.074	0.290	1.000	88.778
liv.dat	0.055	0.630	0.073	0.032	0.107	0.841	0.420	0.500	0.562	0.910	0.368	0.316	1.000	57.500
lng.2c0.dat	0.655	18.432	0.000	0.281	0.969	0.546	0.281	0.500	0.375	0.923	0.250	0.000	1.000	0.571
mag.dat	0.559	2.162	0.081	0.006	0.018	0.717	0.214	0.233	0.284	0.643	0.179	0.232	1.000	1902.000
msh.dat	0.038	11.452	0.000	0.201	0.419	0.491	0.043	0.068	0.003	0.018	0.000	0.064	0.840	369.273
nrs.2c0.dat	0.002	9.722	0.500	1.000	1.000	0.667	0.000	0.000	1.543e-04	0.000	0.000	0.000	1.000	1620.000
opt.2c0.dat	4.729	37.658	0.000	0.471	0.865	1.685	0.002	0.003	7.117e-04	0.437	1.779e-04	0.004	0.992	87.812
pas.2c0.dat	1.628	10.856	1.093e-06	0.583	0.972	0.637	0.333	0.500	0.306	0.719	0.194	0.083	1.000	1.636
pbc.2c0.dat	0.511	9.753	2.100e-05	0.016	0.026	0.319	0.095	0.484	0.055	0.198	0.031	0.168	0.986	547.300
pen.2c0.dat	3.475	12.146	0.755	0.109	0.120	1.973	0.014	0.049	0.001	0.171	6.368e-04	0.019	0.929	687.000
pim.dat	0.576	1.911	0.252	0.007	0.022	0.689	0.350	0.500	0.449	0.846	0.293	0.282	0.999	96.000
pos.2c0.dat	0.172	2.418	0.000	0.478	0.733	0.045	0.022	0.500	0.056	0.259	0.044	0.506	1.000	11.250
seg.2c0.dat	1.798	11.943	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.147	0.003	0.033	0.952	121.579
shs.2c0.dat	1.127	9.692	2.705e-05	0.327	1.000	0.642	0.192	0.163	0.327	0.646	0.173	0.231	1.000	2.167
shu.2c0.dat	6.788	6.315	0.005	0.250	0.885	0.748	0.404	0.423	0.365	0.782	0.231	0.096	1.000	2.261
spa.dat	0.347	5.557	2.533e-33	0.091	0.383	0.772	0.394	0.500	0.160	0.426	0.082	0.103	0.966	80.719
spect.dat	0.017	2.114	0.000	0.142	0.210	0.580	0.206	0.500	0.348	0.773	0.270	0.551	1.000	12.136
spectf.dat	0.553	2.197	3.603e-19	0.296	0.993	0.423	0.206	0.500	0.393	0.845	0.292	0.090	1.000	6.068
statlog-sgm.2c0.dat	1.798	12.532	8.627e-14	0.734	1.000	0.484	0.143	0.500	0.010	0.147	0.003	0.033	0.952	121.579
tae.2c0.dat	6.316	0.608	0.559	0.079	0.146	0.653	0.325	0.500	0.397	0.234	0.192	0.354	0.980	30.200
tao.dat	1.395	4.916	0.479	0.360	0.362	0.590	0.163	0.110	0.091	0.157	0.045	0.155	0.397	944.000
thy.2c0.dat	0.256	3.434	0.001	0.186	0.414	0.588	0.302	0.500	0.102	0.321	0.037	0.160	0.912	43.000
tis.dat	0.472	7.253	0.000	0.033	0.234	1.415	0.070	0.084	0.359	0.871	0.269	0.048	1.000	14.428
tnc.dat	-1	0.754	1.000	0.000	0.000	0.448	0.224	0.304	0.682	0.015	0.677	0.500	1.000	733.667
trn.dat	18.000	8.269	0.000	0.500	0.500	0.706	0.000	0.000	0.200	0.505	0.000	0.000	0.800	0.312
veh.2c0.dat	0.185	2.214	5.339e-04	0.037	0.223	0.504	0.251	0.500	0.359	0.686	0.251	0.335	1.000	47.000
vot.dat	0.066	28.884	1.000	0.000	0.000	0.354	0.037	0.010	0.071	0.249	0.051	0.056	1.000	27.188
wav21.2c0.dat	1.182	3.493	0.036	0.123	0.183	1.020	0.141	0.086	0.248	0.804	0.177	0.112	1.000	238.095
wav40.2c0.dat	1.168	3.681	0.015	0.149	0.211	1.000	0.143	0.081	0.290	0.913	0.208	0.064	1.000	125.000
wbcd.dat	3.568	23.285	0.248	0.119	0.232	0.457	0.034	0.012	0.064	0.358	0.049	0.026	0.861	77.667
wdbc.dat	3.405	19.512	5.683e-11	0.517	0.998	0.539	0.049	0.022	0.083	0.589	0.049	0.021	0.995	18.967
whi.2c0.dat	2.188	10.441	1.019e-06	0.222	0.968	0.687	0.381	0.492	0.365	0.905	0.302	0.103	1.000	2.032
win.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.028	0.073	0.507	0.028	0.006	1.000	13.692
wne.2c0.dat	4.290	28.940	3.962e-05	0.764	1.000	0.371	0.056	0.031	0.073	0.507	0.028	0.017	1.000	13.692
wpbc.dat	0.472	3.320	1.422e-06	0.177	0.990	0.485	0.237	0.500	0.434	0.922	0.278	0.242	1.000	6.000
yea.2c0.dat	0.214	0.743	0.056	0.127	0.177	0.624	0.312	0.500	0.450	0.707	0.288	0.341	0.999	185.500
zoo.2c0.dat	0.344	86.450	0.000	0.802	0.802	0.142	0.000	0.000	0.020	0.108	0.000	0.000	0.970	6.312

T. K. Ho, M. Basu, and M. Law. Measures of geometrical complexity in classification problems. In *Data Complexity in Pattern Recognition*, pages 1–23. Springer, 2006.

A. Hoekstra and R. P. W. Duin. On the nonlinearity of pattern classifiers. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 271–275, Washington, DC, USA, 1996. IEEE Computer Society.

A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems in Information Transmission*, 1:1–7, 1965.

F. Lebourgeois and H. Emptoz. Pretopological approach for supervised learning. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 256–260, Washington, DC, USA, 1996. IEEE Computer Society.

M. Li and M. B. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag, Berlin, 1993.

J. M. Maciejowski. Model discrimination using an algorithmic information criterion. *Automatica*, 15:579–593, 1979.

W. Malina. Two-parameter fisher criterion. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 31(4):629–636, 2001.

E. H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:394–395, 1920.

R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208, Cambridge, MA, USA, 1998. MIT Press.

S. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13 (3):252–264, 1991.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12): 1213–1228, 1986. ISSN 0001-0782.

V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

V. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

I. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.