BMC Bioinformatics

Software

Feature selection environment for genomic applications

Fabrício Martins Lopes*[1,2], David Corrêa Martins Jr*[1] and
Roberto M Cesar Jr[1]

Address: [1]Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508-090, São Paulo-SP, Brazil and [2]COINF, Universidade Tecnológica Federal do Paraná, Av. Alberto Carazzai, 1640, 86300-000, Cornélio Procópio-PR, Brazil

Email: Fabrício Martins Lopes* - fabricio@utfpr.edu.br; David Corrêa Martins* - davidjr@vision.ime.usp.br;
Roberto M Cesar - cesar@vision.ime.usp.br

* Corresponding authors

Published: 22 October 2008

BMC Bioinformatics 2008, **9**:451 doi:10.1186/1471-2105-9-451

Received: 30 May 2008
Accepted: 22 October 2008

This article is available from: http://www.biomedcentral.com/1471-2105/9/451

Abstract

Background: Feature selection is a pattern recognition approach to choose important variables according to some criteria in order to distinguish or explain certain phenomena (i.e., for dimensionality reduction). There are many genomic and proteomic applications that rely on feature selection to answer questions such as selecting signature genes which are informative about some biological state, e.g., normal tissues and several types of cancer; or inferring a prediction network among elements such as genes, proteins and external stimuli. In these applications, a recurrent problem is the lack of samples to perform an adequate estimate of the joint probabilities between element states. A myriad of feature selection algorithms and criterion functions have been proposed, although it is difficult to point the best solution for each application.

Results: The intent of this work is to provide an open-source multiplataform graphical environment for bioinformatics problems, which supports many feature selection algorithms, criterion functions and graphic visualization tools such as scatterplots, parallel coordinates and graphs. A feature selection approach for growing genetic networks from seed genes (targets or predictors) is also implemented in the system.

Conclusion: The proposed feature selection environment allows data analysis using several algorithms, criterion functions and graphic visualization tools. Our experiments have shown the software effectiveness in two distinct types of biological problems. Besides, the environment can be used in different pattern recognition applications, although the main concern regards bioinformatics tasks.

Background

Pattern recognition methods allow the classification of objects or patterns in a number of classes [1,2]. In statistical pattern recognition, given a set of classes and an unknown object, a pattern recognition system associates the object to one of the classes, based on defined measures in a feature space. In many applications, specifically in bioinformatics, the feature space dimension tends to be very large, making difficult both classification tasks and network inference. In order to overcome this inconvenient situation, the study of dimensionality reduction problem in pattern recognition is very important.

The so called "curse of dimensionality" [3] is a phenomenon in which the number of training samples required to a satisfactory classification performance or network inference is given by an exponential function of the feature space dimension. This is the main motivation by which performing dimensionality reduction is important in problems with large number of features and small number of training samples. Many applications in bioinformatics belong to this context. Data sets containing mRNA transcription expressions from microarray or SAGE, for example, present thousands of genes (features) and only some dozens of samples that may come from cell states or types of tissues.

In this context, this work presents an open-source feature selection environment, which includes different search algorithms and criterion functions. Due to the curse of dimensionality phenomenon, error estimation is a crucial issue. We have developed two ways to embed error estimation in the criterion functions, one based on classification error and another based on classification information. The main idea is based on the penalization of non-observed or rarely observed instances. After the feature selection, it is possible to apply classical error estimation methods like resubstitution and holdout cross-validation. Illustrative results obtained on gene regulation networks inference and on classification of breast cancer cells using the proposed software are included in the present paper.

Methods

Data preprocessing

A critical issue in gene expression data analysis is the preprocessing of the data. Depending on the criterion function used, a step of data quantization may be required if the original input data is not discrete. The quantization usually requires a normalization of the data. In the software, the *normal transformation* is implemented. It consists in subtracting the signal (expression profile) by its mean and then dividing by its standard deviation. Formally, for every signal $g(t)$, its equation is given by $\eta[g(t)] = \frac{g(t)-E[g(t)]}{\sigma[g(t)]}$, where $E[g(t)]$ and $\sigma[g(t)]$ are, respectively, the expectation and standard deviation of $g(t)$. There are two options: 1 – normalization by feature (g is a feature and t is a sample); 2 – normalization by sample (g is a sample and t is a feature).

The quantization of a normalized signal $\eta[g(t)]$ (g being feature or sample depending on the chosen option for normalization) is a mapping from the continuous expression into k qualitative expressions $\{0, 1, ..., k - 1\}$. It is performed by a threshold mapping that divides the interval given by the minimum and the maximum values of g in k intervals of equal size. A given continuous value r of $\eta[g(t)]$ is mapped to an integer i - 1, where i is the i-th interval to which r belongs.

Implemented feature selection algorithms

The simplest implemented feature selection algorithm is the exhaustive search. This algorithm searches the whole search space. As a result, the selected features are optimal according to the criterion function chosen to guide the algorithm. However, the computational cost often makes this approach inadequate, thus requiring alternative (non-optimal) algorithms.

In this work we have implemented two sub-optimal approaches with unique solution, being also known as wrappers [4]. In the first one, the selected subset starts empty and features are inserted by optimizing a criterion function until a stop condition is satisfied, which is often based on the subset size or a threshold. This method is known as Sequential Forward Selection (SFS) [5]. Although simple and efficient, SFS presents an undesirable drawback known as nesting effect. This effect arises because the selected features are never discarded.

In order to circumvent the nesting effect, the Sequential Floating Forward Selection (SFFS) is also available in the software. The SFFS algorithm may be formalized as in [5]. A schematic flowchart of the SFFS algorithm is presented in Figure 1. In this algorithm, the SFS and the SBS (Sequential Backward Search – a counterpart of the SFS) are successively applied.

Considering the adopted feature selection and criterion function, the intermediate results are stored in a list. The best sets of this list are displayed as the algorithm result.

Other feature selection methods will be implemented, both wrappers (e.g., "plus l-take away r", simulated annealing, genetic algorithm) and filters (e.g., correlation-based, consistency-based, entropy-based), and embedded methods (e.g., CART) [3,4,6,7]. Also, methods for feature extraction, such as PCA and ICA [8,9], will be implemented. We hope that the availability of the software as open-source will help the inclusion of other functions by the international community.

Implemented criterion functions

We implemented criterion functions based on classification information (mean conditional entropy) and classification error (Coefficient of Determination [10]), introducing some penalization on poorly or non-observed patterns. Introducing penalization on rarely observed instances embeds the error estimation into the criterion functions. This approach has two main advantages. First, the selected features are obtained not only by

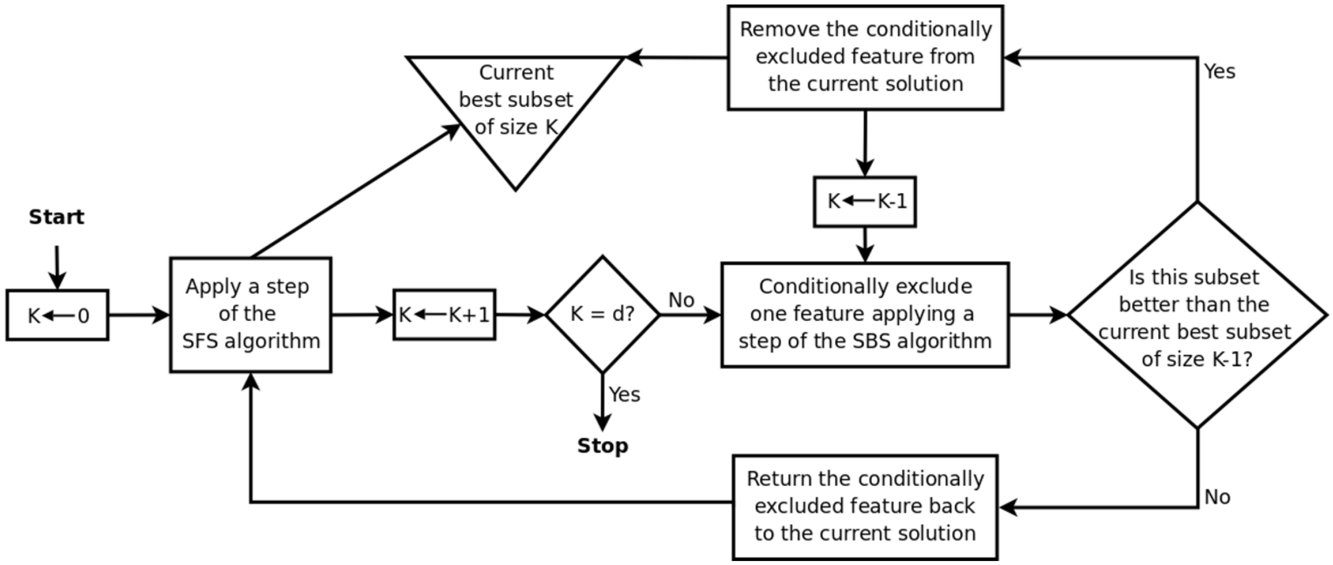

Remove the conditionally excluded feature from the current solution
Yes
Current best subset of size K
K←K-1
Start
Is this subset better than the current best subset of size K-1?
Apply a step of the SFS algorithm
K←K+1
K = d?
No
Conditionally exclude one feature applying a step of the SBS algorithm
K←0
Yes
Stop
Return the conditionally excluded feature back to the current solution
No

Figure 1
Flowchart of the SFFS algorithm. Simplified flowchart of the SFFS algorithm (adapted from [21]).

their informative content about the classes, but also by the number of well observed instances they have (large number of well observed instances means small error estimation). Another advantage is that the right dimension of the feature subset is also estimated (the dimension parameter is not required). In the software, the implemented feature selection algorithms (SFS and SFFS) requires the dimension parameter just to limit the execution time, performing the procedures until the subset size reaches that parameter. According to the criterion function values, the resulting subsets can have smaller dimension than the specified by the parameter.

Mean conditional entropy

Information Theory, originated by Shannon's works [11], can be employed on feature selection problems [2]. The mean conditional entropy of Y given all the possible instances $\mathbf{x} \in \mathbf{X}$ is given by:

$$H(Y \mid \mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} P(\mathbf{x})\, H(Y \mid \mathbf{x}), \tag{1}$$

where $P(Y \mid \mathbf{x})$ is the conditional probability of Y given the observation of the instance $\mathbf{x}$, $H(Y \mid \mathbf{x}) = -\Sigma_{y \in Y} P(y \mid \mathbf{x}) \log P(y \mid \mathbf{x})$. Shannon's entropy can be generalized by using Tsallis Entropy [12] with a parameter q. We have that $q = 1$ is the Shannon's Entropy (extensive entropy), $q < 1$ is the sub-extensive entropy and $q > 1$ is the super-extensive entropy. The Tsallis Entropy is given by $H(Y \mid \mathbf{x}) = \frac{1}{q-1}(1 - \sum_{y \in Y} (P(Y \mid \mathbf{x}))^q)$ Lower values of H yield better feature subspaces (the lower H, the larger is the information gained about Y by observing $\mathbf{X}$).

Coefficient of Determination

The Coefficient of Determination (CoD) [10], like the conditional entropy, is a non-linear criterion useful for feature selection problems [13]. The CoD is given by:

$$\mathrm{CoD}_Y(\mathbf{X}) = \frac{\varepsilon_Y - \varepsilon_Y(\mathbf{X})}{\varepsilon_Y}, \tag{2}$$

where $\varepsilon_Y = 1 - \max_{y \in Y} P(y)$ is the prior error, i.e., the error by predicting Y in the absence of other observations, and $\varepsilon_Y(\mathbf{X}) = \Sigma_{\mathbf{x} \in \mathbf{X}} P(\mathbf{x})\ (1 - \max_{y \in Y} P(y \mid \mathbf{x}))$ is the average error by predicting Y based on the observation of $\mathbf{X}$. Larger values of CoD yield to better feature subspaces (CoD = 0 means that the feature subspace does not improve the prior error and CoD = 1 means that the error is fully eliminated).

Penalization of non-observed instances

In order to embed the error estimation, due to feature vectors with large dimensions and insufficient number of samples, we adopted a strategy to involve non-observed instances in the criterion value calculus [14]. A positive probability mass is attributed to the non-observed instances and the corresponding contribution is the same as observing only the Y values with no other observations. In the case of mean conditional entropy, the non-observed instances receive the entropy equal to $H(Y)$ whereas the CoD receives the prior error ε_Y value. The probability mass for the non-observed instances is parameterized by α. This parameter is added to the absolute frequency (number of occurrences) of all possible instances. So, the mean conditional entropy with this type of penalization becomes:

$$H(Y \mid \mathbf{X}) = \frac{\alpha(M-N)H(Y)}{\alpha M+s} + \frac{\sum_{i=1}^{N}(f_i+\alpha)H(Y|\mathbf{X}=\mathbf{x_i})}{\alpha M+s}, \quad (3)$$

where M is the number of possible instances of the feature vector $\mathbf{X}$, N is the number of observed instances (so, the number of non-observed instances is given by M - N), f_i is the absolute frequency (number of observations) of $\mathbf{x_i}$ and s is the number of samples.

The CoD becomes:

$$\mathrm{CoD}_Y(\mathbf{X}) = 1 - \frac{\alpha(M-N)}{\alpha M+s} - \frac{\sum_{i=1}^{N}(f_i+\alpha)(1-\max_{y\in Y} P(y|\mathbf{x_i}))}{(\alpha M+s)\varepsilon_Y}. \quad (4)$$

Penalization of rarely observed instances

In this penalization, the non-observed instances are not taken into account. This penalization consists in changing the conditional probability distribution of the instances that have just a unique observation [15]. It makes sense because if an instance $\mathbf{x}$ has only 1 observation, the value of Y is fully determined ($H(Y \mid \mathbf{X} = \mathbf{x}) = 0$ and $\mathrm{CoD}_Y(\mathbf{X}) = 1$), but the confidence about the real distribution of $P(Y \mid \mathbf{X} = \mathbf{x})$ is very low. A parameter β gives a confidence value that $Y = y$. The main idea is to equally distribute 1 - β over all $P(Y \neq y \mid \mathbf{X} = \mathbf{x})$ and to attribute β to $P(Y = y \mid \mathbf{X} = \mathbf{x})$. In Barrera *et al* [15], the β value is $\frac{1}{|Y|}$ where $|Y|$ is the number of classes (cardinality of Y), i.e., the uniform distribution (strongest penalization). By adapting this penalization to the Equation (1), the mean conditional entropy becomes:

$$H(Y \mid \mathbf{x}) = \frac{M-N}{s} H((F(0),\ldots,F(|Y|-1))) + \sum_{\mathbf{x}\in \mathbf{X}:P(\mathbf{x})>\frac{1}{s}} P(\mathbf{x})\, H(Y \mid \mathbf{x}), \quad (5)$$

where $F: \{0, 1, \ldots, |Y| - 1\} \rightarrow [0, 1]$ is the probability distribution given by

$$F(i) = \begin{cases} \beta & \text{for } i = y, \\ \frac{1-\beta}{|Y|-1} & \text{for } i \neq y, \end{cases}$$

and N is the number of instances $\mathbf{x}$ with $P(\mathbf{x}) > \frac{1}{s}$ (more than one observation).

Since $\varepsilon_Y(\mathbf{x}) = 1 - \beta$ when $P(Y \mid \mathbf{x}) = \frac{1}{s}$, the CoD with this penalization is given by:

$$\mathrm{CoD}_Y(\mathbf{X}) = 1 - \frac{(M-N)(1-\beta)}{s\varepsilon_Y} - \frac{\sum_{\mathbf{x}\in \mathbf{X}:P(\mathbf{x})>\frac{1}{s}} P(\mathbf{x})(1-max_{y\in Y}P(y|\mathbf{x}))}{\varepsilon_Y}. \quad (6)$$

Classifier design and generalization

After the feature selection, the classifier is designed from the table of conditional probabilities of the classes (Y) given the patterns ($\mathbf{X}$). This table is used as a Bayesian classifier where, for each given instance, the chosen label is the one with maximum conditional probability for the considered instance. In case of ties, i.e., instances that have two or more labels of maximum probability (including non-observed instances), generalization according to some criterion is carried out. A commonly used criterion is the nearest neighbors with some distance metric [1,2]. The nearest neighbors using Euclidean distance has been implemented for generalization. In this implementation, the nearest neighbors with equal distances are taken and the occurrences of each label given these patterns are summed. If the tie persists, the next nearest neighbors are taken and the process is repeated until only one of such labels has the maximum number of occurrences (i.e., until the tie is broken). The label with the maximum number of occurrences at the end of this process is chosen as the class to which the considered instance belongs.

There are many classifiers described in the literature, such as Parzen, Fisher Linear Discriminant, Binary Decision Tree, Perceptron, Support Vector Machines and so on [3]. The software has been designed so that such additional classifiers may be incorporated. One of the objectives of the software is to implement as many classifiers as possible. However, the primary focus is given on feature selection algorithms and criterion functions.

Error estimation methods

In order to validate and compare feature selection algorithms, error estimation methods are required, especially for data with high dimensionality and small number of samples which is usually the case in bioinformatics problems.

The first method implemented in the software is the simple resubstitution in which the training set is also used as testing set. For many bioinformatics problems, the resubstitution error estimation works fine since usually there is not enough data samples to perform a good estimation, and partitioning the data leads to an even smaller training sample size (see [16] for a detailed discussion).

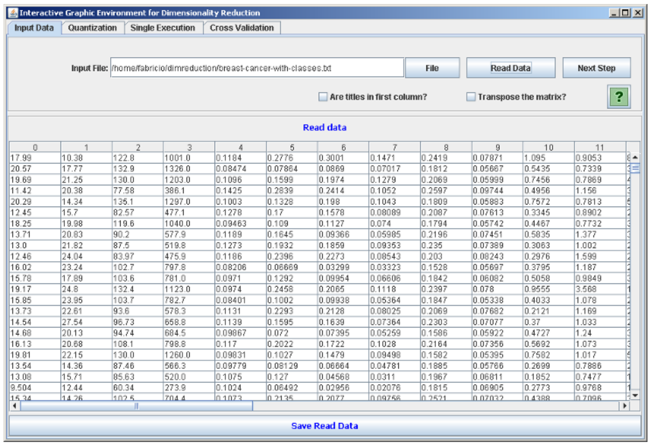
The Holdout cross validation is also available in the software. It consists in partitioning the data in two sets: training and test sets. The size of the training set can be defined in terms of percentage of total data samples. Thus, the feature selection method is applied to the training set, the classifier is designed taking into account the resulting feature set and the conditional probability distributions given by the training set. Finally, the designed classifier is applied to the test set and the classification error rate is obtained. Alternative error estimation methods like K-fold cross-validation, leave-one-out (a particular kind of cross-validation) and bootstrap will be implemented in the software.

Network inference

If one considers the training set as time series data, i.e., each sample representing a state of the system at time t, it is possible to generate networks of prediction based on the Probabilistic Genetic Network model described in [15]. This model is a Markov chain that mimics the properties of a gene as a non-linear stochastic gate. Given a fixed gene Y as target, the general idea of this model is to determine the subset of genes $\mathbf{Z} \subseteq \mathbf{X}$ that makes the best prediction of the Y value in the next instant of time. In order to achieve this intent, the system is considered translation invariant, i.e., the transition function is the same for every time instant t. In this way, it is possible to apply a feature selection method for every target, since a sample is represented by the pair $(\mathbf{X}[t], Y[t + 1])$ and the training set is composed by a collection of these samples along the time domain (so, the number of samples is given by t - 1). After the application of the feature selection process for every target, the best feature subsets obtained for each target gene define the network connections.

Software description

The software is implemented in Java in order to be executable in different platforms. It is open source and intended to be continuously developed in a world-wide collaboration. There are four main panels: the first one allows the user to load the data set (Figure 2). The second is optional for the user to define a quantization degree to the data set. The quantized data may be visualized (Figure 3). It is worth noting that the mean conditional entropy and the CoD require data quantization to discrete values. This fact explains the quantization step available in the software. Please refer to the Section *Data Preprocessing* for further information on the normalization and quantization methods applied. The reason to keep the quantization optional is because other criterion functions, such as the distance-based criteria, do not require quantization and the software is intended to implement as many criterion functions as possible. The third panel is dedicated to perform single tests (Figure 4). With this panel, the user can execute feature selection methods (see Section *Implemented feature selection algorithms*) along with the criterion function (see Section *Implemented criterion functions*). The user has two options: the first one is to apply the simple resubstitution method (i.e., to use the same input data as



Input File: /home/fabricio/dimreduction/breast-cancer-with-classes.txt

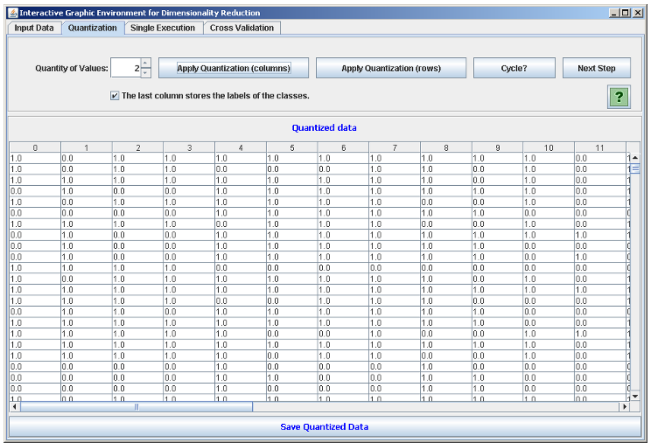
□ Are titles in first column? □ Transpose the matrix?

Read data

0	1	2	3	4	5	6	7	8	9	10	11
17.99	10.38	122.8	1001.0	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053
20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339
19.69	21.25	130.0	1203.0	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869
11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156
20.29	14.34	135.1	1297.0	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813
12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902
18.25	19.98	119.6	1040.0	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732
13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377
13.0	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002
12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976	1.599
16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	0.3795	1.187
15.78	17.89	103.6	781.0	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058	0.9849
19.17	24.8	132.4	1123.0	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	0.9555	3.568
15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	0.4033	1.078
13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	0.2121	1.169
14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	0.37	1.033
14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922	0.4727	1.24
16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	0.5692	1.073
19.81	22.15	130.0	1260.0	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	0.7582	1.017
13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	0.2699	0.7886
13.08	15.71	85.63	520.0	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	0.1852	0.7477
9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	0.2773	0.9768
15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032	0.4388	0.7096

Figure 2

Upload biological data. The software panel to upload biological data.


Interactive Graphic Environment for Dimensionality Reduction
Input Data
Quantization
Single Execution
Cross Validation
Apply Quantization (columns)
Apply Quantization (rows)
Cycle?
Next Step
?
Save Quantized Data

Quantity of Values: 2

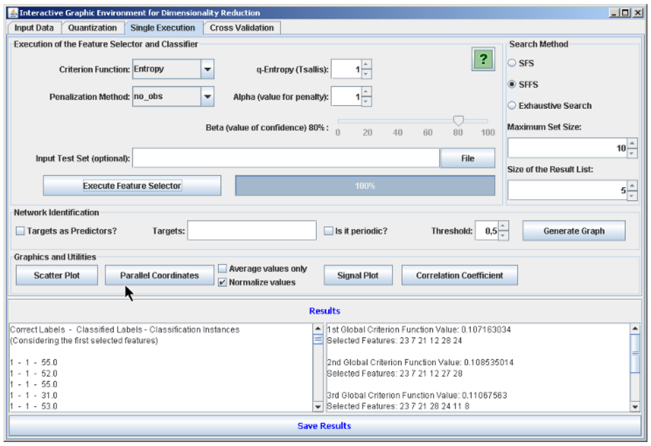
☑ The last column stores the labels of the classes.

Quantized data

0	1	2	3	4	5	6	7	8	9	10	11
1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0
0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0
1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0
0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0

Figure 3

Quantization process. The software panel to apply quantization process.



Criterion Function: Entropy
q-Entropy (Tsallis): 1
Penalization Method: no_obs
Alpha (value for penalty): 1
Beta (value of confidence) 80% : 0 20 40 60 80 100
Input Test Set (optional):
○ SFS
● SFFS
○ Exhaustive Search
Maximum Set Size: 10
Size of the Result List: 5
□ Targets as Predictors?
Targets:
□ Is it periodic?
Threshold: 0,5
□ Average values only
☑ Normalize values

```
Correct Labels  -  Classified Labels - Classification Instances
(Considering the first selected features)

1  -  1  -  55.0
1  -  1  -  52.0
1  -  1  -  55.0
1  -  1  -  31.0
1  -  1  -  53.0
```

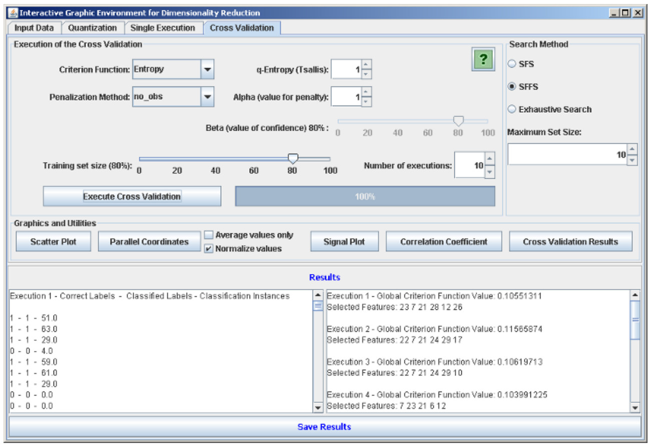
```
1st Global Criterion Function Value: 0.107163034
Selected Features: 23 7 21 12 28 24

2nd Global Criterion Function Value: 0.108535014
Selected Features: 23 7 21 12 27 28

3rd Global Criterion Function Value: 0.11067563
Selected Features: 23 7 21 28 24 11 8
```

Figure 4

Single feature selection execution. The software panel to apply single feature selection execution and feature selection approach for growing of genetic networks from target genes.

training and test set); the second option consists in to apply the classifier on a different data set given by the user specified at input test set text box (Holdout cross-validation with training and test sets partitioning defined by the user).

The fourth panel is very similar to the previous one, but allows the application of automatic cross-validation procedures [17] (Figure 5) (see Section *Error estimation methods*). This panel presents only the Holdout cross-validation with training and test sets partitioning performed randomly.



Criterion Function: Entropy
q-Entropy (Tsallis): 1
Penalization Method: no_obs
Alpha (value for penalty): 1
Beta (value of confidence) 80% : 0 20 40 60 80 100
Training set size (80%): 0 20 40 60 80 100
Number of executions: 10
○ SFS
● SFFS
○ Exhaustive Search
Maximum Set Size: 10
□ Average values only
☑ Normalize values

```
Execution 1 - Correct Labels  -  Classified Labels - Classification Instances

1  -  1  -  51.0
1  -  1  -  63.0
1  -  1  -  29.0
0  -  0  -  4.0
1  -  1  -  59.0
1  -  1  -  61.0
1  -  1  -  29.0
0  -  0  -  0.0
0  -  0  -  0.0
```

```
Execution 1 - Global Criterion Function Value: 0.10551311
Selected Features: 23 7 21 28 12 26

Execution 2 - Global Criterion Function Value: 0.11565874
Selected Features: 22 7 21 24 29 17

Execution 3 - Global Criterion Function Value: 0.10619713
Selected Features: 22 7 21 24 29 10

Execution 4 - Global Criterion Function Value: 0.103991225
Selected Features: 7 23 21 6 12
```

Figure 5

Cross-validation. The software panel to apply feature selection execution using cross-validation.

There are other implemented utilities, including the visualization of the results of the feature selection as scatter-plots, parallel coordinates [18] and graphs. Every panel has its own help button that activates a popup window explaining the functionalities and options regarding the corresponding panel. Also, description boxes are displayed when the mouse cursor is placed over some option in order to quickly recall the user about the functionalities of the system.

Results and discussion

This section presents some illustrative results in two different typical situations in bioinformatics. All results shown here were obtained by using the default values in the software: SFFS with mean conditional entropy and penalization of non-observed instances, α = 1, Tsallis parameter q = 1, and maximum set size = 10.

Initially the software was applied for feature selection in a biological classification problem to classify breast cancer cells [19] in two possible classes: benign and malignant. The training set is composed by 589 instances and 32 features. The SFFS algorithm with the entropy criterion was explored. The results shown in Figure 6 present very low variations and high accurate classification achieving 96.30% of accuracy on average. The scatterplot taking the first two features of the feature set result obtained by SFFS is shown in Figure 7, while the parallel coordinates of the resulting six features are shown in Figure 8.

The second addressed computational biology problem is genetic network identification. In this case we used an artificial genetic network generated by the approach presented in [20]. The adopted parameters were: 10 nodes, binary quantization, 20 transitions (instants of time), 1 average degree per node and random graphs of Erdös-Rényi. Figure 9 presents the network identification, in which no false negatives occurred and just few false positives. The methodology of network generation is described in Section *Network inference*, the same as used to generate probabilistic genetic networks of the *Plasmodium falciparum* from microarray data [15], thus showing the possibility of exploring the software in such an important problem in bioinformatics.

Conclusion

The proposed feature selection environment allows data analysis using several algorithms, criterion functions and graphic visualization tools. Our experiments have shown the software effectiveness in two distinct types of biological problems. Besides, the environment can be used in different pattern recognition applications, although the main concern regards bioinformatics tasks, especially those involving high-dimensional data (large number of genes, for example) with small number of samples. Users

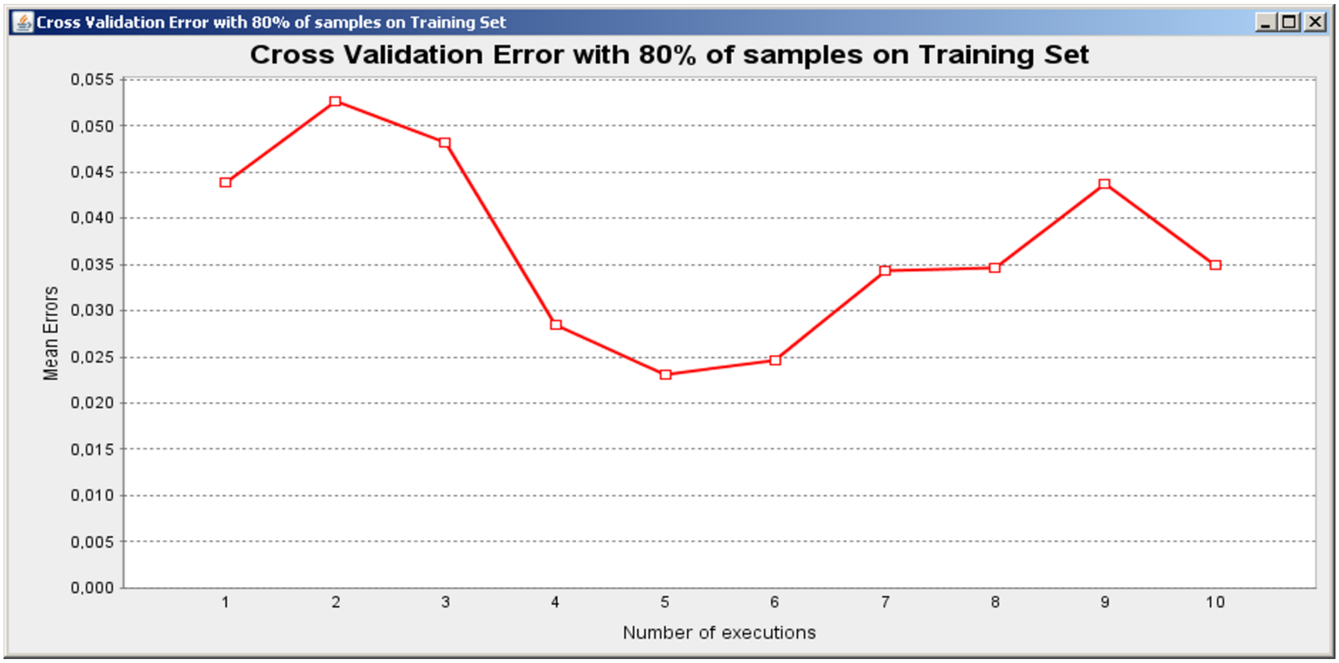


Figure 6
Cross-validation results. Illustrative cross-validation results using 10 executions, 80% of data as training set and 20% as test set.

without programming skills are allowed to manipulate the software in an easy way, just by clicking to select file inputs, quantization, algorithms, criterion functions, error estimation methods and visualization of the results, while an intuitive help system quickly presents the instructions of the present functionalities that the user may look for. On the other hand, the availability of the software as open-source allows programmers to explore the implemented methods as libraries in their own programs. The environment is implemented in "wizard style", i.e., with tabs delimiting each procedure. A complete technical report that complements the help information is available at the project web site. Two video demos

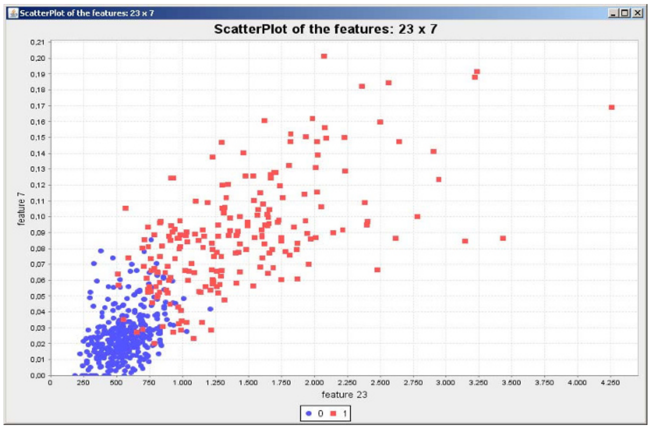


Figure 7
Scatterplot generated over the benign and malignant tissues data set. Scatterplot generated by using the first two features included in the SFFS result over the benign and malignant tissues data set.

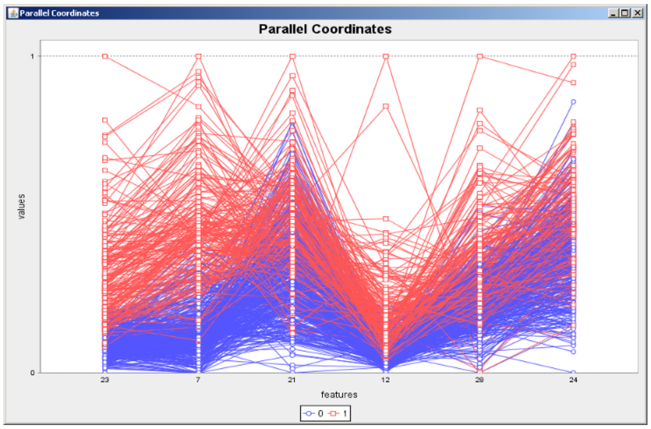


Figure 8
Parallel coordinates generated over the benign and malignant tissues data set. Parallel coordinates generated by using the six features included in the SFFS result over the benign and malignant data set. The original values were normalized between 0 and 1.

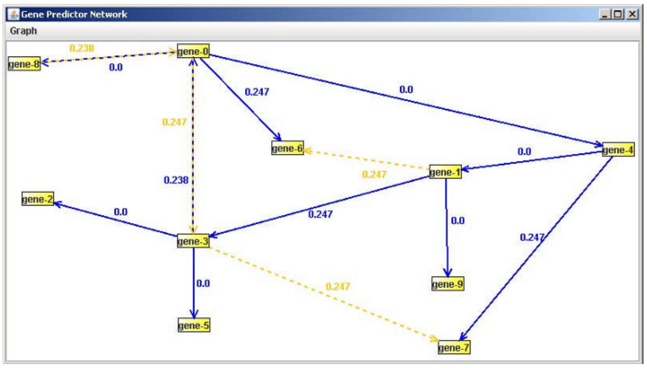


Figure 9
Results of the growing genetic networks from target genes. Identified network: dashed lines represent the false positives and solid lines the positives. There are no false negatives.

of the software in use are available at http://dimreduction.incubadora.fapesp.br/.

This software opens space for future work. The next steps consist in the implementation of other classical feature selection algorithms, including wrappers, filters and embedded methods, criterion functions (e.g., based on distances between classes [1]); classifier design methods like Parzen, Fisher Linear Discriminant, Binary Decision Tree, Perceptron and Support Vector Machines; error estimation methods such as leave-one-out, K-fold cross-validation and bootstrap; and the inclusion of classical methods of feature extraction, such as PCA and ICA [8,9].

Availability and requirements

• Project name: DimReduction

• Project home pages: http://dimreduction.incubadora.fapesp.br and http://sourceforge.net/projects/dimreduction

• Operating system: Platform independent

• Programming language: Java

• Other requirements: Java Runtime Environment (JRE) 1.5 or higher

• License: code available; GNU Lesser General Public License

• Video demos: http://dimreduction.incubadora.fapesp.br

• Complete technical report: http://arxiv.org/abs/0805.3964

Authors' contributions

FML and DCM wrote the software code, analyzed the data and wrote the manuscript. RMCJ participated in the design and coordination of the study. All authors contributed to, read and approved the final manuscript.

Acknowledgements

This work was supported by FAPESP, CNPq and CAPES.

References

1. Theodoridis S, Koutroumbas K: *Pattern Recognition* 1st edition. USA: Academic Press; 1999.
2. Duda RO, Hart PE, Stork DG: *Pattern Classification* 2nd edition. NY: Wiley-Interscience; 2000.
3. Jain AK, Duin RP, Mao J: **Statistical Pattern Recognition: A Review.** *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000, **22:**4-37.
4. Guyon I, Elisseeff A: **An introduction to variable and feature selection.** *Journal of Machine Learning Research* 2003, **3:**1157-1182.
5. Pudil P, Novovičová J, Kittler J: **Floating search methods in feature selection.** *Pattern Recogn Lett* 1994, **15(11):**1119-1125.
6. Hall MA, Smith LA: **Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper.** *Proc FLAIRS Conference* 1999:235-239 [http://www.aaai.org/Library/FLAIRS/1999/flairs99-042.php]. AAAI Press
7. Kohavi R, John G: **Wrappers for feature selection.** *Artificial Intelligence* 1997, **97:**273-324.
8. Jolliffe IT: *Principal component analysis* New York: Springer-Verlag; 1986.
9. Comon P: **Independent Component Analysis, a New Concept?** *Signal Processing* 1994, **36(3):**287-314.
10. Dougherty ER, Kim S, Chen Y: **Coefficient of determination in nonlinear signal processing.** *Signal Processing* 2000, **80:**2219-2235.
11. Shannon CE: **A mathematical theory of communication.** *Bell System Technical Journal* 1948, **27:**379-423 [http://plan9.bell-labs.com/cm/ms/what/shannonday/]. 623–656
12. Tsallis C: *Nonextensive Statistical Mechanics and its Applications* Lecture Notes in Physics, Berlin: Springer; 2001.
13. Hsing T, Liu L, Brun M, Dougherty ER: **The coefficient of intrinsic dependence (feature selection using el CID).** *Pattern Recognition* 2005, **38(5):**623-636.
14. Martins DC Jr, Cesar RM Jr, Barrera J: **W-operator window design by minimization of mean conditional entropy.** *Pattern Analysis & Applications* 2006, **9:**139-153.
15. Barrera J, Cesar RM Jr, Martins DC Jr, Vencio RZN, Merino EF, Yamamoto MM, Leonardi FG, Pereira CAB, Portillo HA: *Methods of Microarray Data Analysis V, Springer-Verlag 2006 chap. Constructing probabilistic genetic networks of Plasmodium falciparum from dynamical expression signals of the intraerythrocytic development cycle* [http://www.springer.com/life+sci/bioinformatics/book/978-0-387-34568-0].
16. Braga-Neto U, Dougherty ER: **Is cross-validation valid for small-sample microarray classification?** *Bioinformatics* 2004, **20(3):**374-380.
17. Kohavi R: **A study of cross-validation and bootstrap for accuracy estimation and model selection.** *Proceedings of the 14th International Joint Conference on Artificial Intelligence* 1995, **2:**1137-1145.
18. Inselberg A: **The Plane with Parallel Coordinates.** *Journal The Visual Computer* 1985, **1(4):**69-91.
19. Asuncion A, Newman D: **UCI Machine Learning Repository.** 2007 [http://mlearn.ics.uci.edu/MLRepository.html].
20. Lopes FM, Cesar RM Jr, da F Costa L: **AGN Simulation and Validation Model.** In *Advances in Bioinformatics and Computational Biology, Third Brazilian Symposium on Bioinformatics Volume 5167*. LNBI, Springer; 2008:169-173.
21. de Campos T: **Técnicas de Seleção de Características com Aplicações em Reconhecimento de Faces.** In *Master's thesis* University of Sao Paulo; 2001. [Awarded as the Best Computer Science MSc. Thesis in the CTD/SBC 2002]