A Platform for the Selection of Genes in DNA Microarray Data using Evolutionary Algorithms

Miguel Rocha
CCTC/ Dep. Informatics
Universidade do Minho
Campus Gualtar, Braga, Portugal
mrocha@di.uminho.pt

Rui Mendes
CCTC/ Dep. Informatics
Universidade do Minho
Campus Gualtar, Braga, Portugal
rcm@di.uminho.pt

Paulo Maia
CCTC/ Dep. Informatics
Universidade do Minho
Campus Gualtar, Braga, Portugal
paulo.maia@di.uminho.pt

Daniel Glez-Peña
ESEI/ Computer Science Dep.
University of Vigo
Campus Universitario As Lagoas s/n, Ourense, Spain
dgpena@uvigo.es

Florentino Fdez-Riverola
ESEI/ Computer Science Dep.
University of Vigo
Campus Universitario As Lagoas s/n, Ourense, Spain
riverola@uvigo.es

ABSTRACT

This paper presents a flexible framework to the task of feature selection in classification of DNA microarray data. The user can select a number of filter methods in the preprocessing stage and choose from a wide set of classifiers (models and algorithms from WEKA [17] are available) and accuracy estimation methods. This approach implements wrapper methods, where *Evolutionary Algorithms*, with variable-sized set based representations are used to reduce the number of attributes. Two case studies were used to validate the approach, with three distinct classifiers (1-nearest neighbour, decision trees, SVMs), a filter method based on *discriminant fuzzy patterns* and *k-fold cross-validation* to estimate the generalization error.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES—*Biology and Genetics*; H.2.8 [**Information Systems**]: DATABASE MANAGEMENT—*Database Applications/Data mining*

General Terms

Algorithms, Experimentation

1. INTRODUCTION

The application of *Machine Learning* techniques in the context of DNA microarray data is becoming quite important in the biomedical research. In particular, the automatic classification of samples has been a promising approach in cancer diagnosis[3] and a number of classifiers have been proposed, including *Support Vector Machines (SVMs)* [14], *Neural Networks (NNs)* [9] or *k-nearest neighbor (kNN)* methods [8]. A major problem with the application of these methods is the huge number of attributes (genes) in the datasets (typically thousands). Gene reduction in microarray data is extremely important because it usually increases the accuracy of the machine learning techniques and it provides clues to researchers about genes that are important in a given context (e.g. biomarkers for certain diseases).

There are two approaches for feature selection: filters and wrappers. Filters are applied in the preprocessing stage, using some measure of relevance and are independent of the learning algorithm used. These methods may overlook relationships among genes and prune genes that, by themselves, seem unimportant but that may explain the phenomena studied, when taken in consideration together with others.

On the other hand, wrappers train the classifier with a gene subset and estimate its generalization error. These methods do not have the possible shortcomings of filters but it is important to ensure that they do not overfit the data. Besides, they are dependent of the classifier that is used. Indeed, there is no guarantee that an optimal subset of genes chosen for one classifier, will be the optimal one when used with another algorithm.

This paper presents a flexible framework for the task of gene selection in classification of DNA microarray data. The user is free to apply any number of preprocessing filters she deems necessary prior to the task. These will usually be based on statistical measures. As the framework interacts with the WEKA software [17], it is possible to use any classifier it provides. Thus, this framework may work as a wrapper to select genes for any of the classifiers provided by WEKA.

In the wrapper method, an *Evolutionary Algorithm (EA)* was developed to provide the optimization engine. The *EA* uses a variable-sized set-based representation to encode the set of genes to use by the classifier. The fitness function of the *EA* is computed by taking into account an accuracy estimation of the classifier, but can take other metrics into account (e.g. number of genes used).

It is important not to confuse the fitness function with the evaluation of the generalization error of the solutions produced by the *EA* at the end of the run. In this framework, the training set is used by the fitness function to perform the 5-fold cross-validation. The evaluation of the solutions provided by the *EA* at the end of the run are tested on a test set that it never saw during evolution. To provide a less biased estimation of the generalization error, a 5 times 10-fold cross-validation scheme was used.

2. PREVIOUS RESEARCH

Some researchers use the test set as a validation set, i.e., it is provided to the *EA* when computing the accuracy measure included in the fitness function The error rate estimate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model and thus overfitting can occur over this data. If for some reason a validation set is used in training to evaluate the generalization capabilities of the solutions, it is imperative to use a test set with a set of examples used *only* to assess the performance of the trained classifier.

There are several researchers that have used *EAs* for gene selection in microarray data in previous work. Li et al [10] used an *EA* to select an arbitrarily fixed set of 50 genes and used a consensus *kNN* method for classification. The validation of their work uses a single test set for each data set (the holdout method). Keedwell et al [9] use a *NN* as a classifier. They only used a training set and didn't provide any kind of estimation of the generalization error. Deutsch et al [3] describes a replication algorithm with a *kNN* classifier ($k = 1$). The holdout method was only used in some of the datasets.

Ooi et al [13] used an *EA* with a maximum likelihood classifier. Their fitness function uses the error of the test set. This makes it useless for purposes of estimation of the generalization method, as the *EA* was trained with it. Peng and Liu [14, 11] present similar approaches. Their EA is based on Ooi's approach and they use *SVMs* as a classifier. They use leave-one-out cross-validation as a fitness function and present that result as an estimation of the generalization method. That is not a valid estimation as was explained before. Umpai et al [8] uses an *EA* and a *kNN* as a classifier and the holdout method for estimating the generalization method.

Fröhlich et al [5] describe an *EA* used with a *SVM*. Their approach allows the selection of either a given number of genes or a variable one. They provide *k-fold cross-validation* like estimation of the generalization error.

Most of these approaches did not provide acceptable estimations of the generalization error. This is extremely important in the field of Machine Learning and only k-fold cross-validation (or leave-one-out if the number of examples is too small) is acceptable. Some of the approaches are limited to an arbitrarily fixed number of genes and all of the approaches only use a single classifier and there is no simple way to implement other alternatives.

3. THE PROPOSED APPROACH

In this section, the proposed approach to feature selection and classification of DNA microarrays data is described. The overall system aims at taking a given classification dataset with DNA microarray data and provide, as the final outcome, two main results: a list of genes that can discriminate between the classes and a final classifier that is able to predict the class of new examples. The process is organized as a flow of data blocks that are sequentially processed by several components, thus defining a processing flow where the outputs of some processes are taken as the inputs to others to achieve the desired results at the end.

The main components that integrate this tool are illustrated in Figure 1 and can be briefly summarized:

- *Feature selection - filter methods*: this component is able to apply a number of filters over the dataset, that can reduce its number of attributes (genes). These methods are independent of the classifier that will be used and are typically obtained by some statistical calculations over the data.

- *Classifier - error estimation*: this component handles the estimation error of a classifier in a given dataset, i.e. it implements the process of choosing a class of classifiers and an error estimation method (e.g. cross validation, holdout). The outcome is an estimate of the accuracy of the classifier in the dataset.

- *Feature selection - wrapper methods*: this component is closely connected to the previous, implementing an optimization procedure that tries to find the best solution to the feature selection problem (i.e. the best list of genes). The cost function for the optimization method takes into account the error estimates that are provided by the previous module.

- *Build classifier*: this module handles the construction of the final classifier given the dataset filtered by the two feature selection schemes.

3.1 Filter methods

A number of methods can be used to filter a subset of genes from a microarray dataset that are independent of the classification model. These genes are expected to be up- or down-regulated between healthy and diseased tissues or between different classes. A differentially expressed gene is a gene which has the same expression pattern for all samples of the same class, but different for samples belonging to different classes. The relevance value of a gene depends on its ability to be differentially expressed. However, a non-differentially expressed gene will be considered irrelevant and will be removed from a classification process even though it might well contain information that would improve classification accuracy. One way or another, the selected method has to pursue two main goals: (i) reduce the cost and complexity of the classification model and (ii) improve its accuracy.

Some simple schemes of filtering *flat* genes can be used by computing a standard deviation or analyzing the peaks in the gene expression. More complex methods include the use of information gain metrics in order to rank the genes [1], redundancy reduction and feature extraction [7, 15], as well as the identification of similar gene classes making prototypes-genes [6] or the use of *Markov blanket filters* [18], just to name a few.

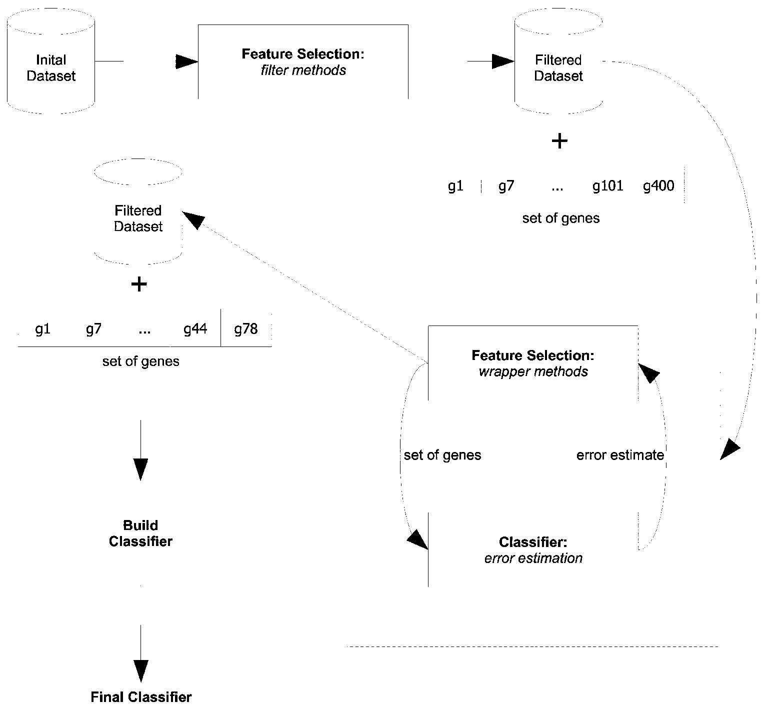

Figure 1: Main components of this tool.

These methods rank genes depending on their relevance for discrimination. Then by setting a threshold, one can filter the less relevant genes among those considered. As such, these filtering methods may be seen as particular gene selection methods. An important task in microarray data analysis is therefore to identify genes, which are differentially expressed in this way. Statistical analysis of gene expression data relating to complex diseases is of course not really expected to yield accurate results. A realistic goal is to narrow the field for further analysis, to give geneticists a short-list of genes for analysis into which hard-won funds are worth investing. The aim of this component is to allow the implementation of any filter method and therefore a flexible architecture was designed to allow the definition of new methods and plug them into the system. In the experiments provided in this work, a method for selecting genes based on the notion of *fuzzy pattern*, is used.

The whole algorithm comprises of three main steps. First, we represent each gene value in terms of one from the following linguistic labels: *Low*, *Medium* and *High* and their intersections *LowMedium* and *MediumHigh*. The output is a *fuzzy microarray descriptor* (FMD) for each existing sample (microarray). The second phase aims to find all genes that best explain each pathology, constructing a supervised *fuzzy pattern* (FP) for each class. Starting from the previous obtained FPs, our proposed method is able to discriminate those genes that can provide a substantial discernibility between existing classes, generating an unique *discriminant fuzzy pattern* (DFP).

3.1.1 *Discretizing microarray data using fuzzy labels (FMD)*

Given a set of n features or attributes (in this work, gene expression levels), $\boldsymbol{F} = \{F_1, F_2, ..., F_n\}$, the discretization process is based on determining the membership function of each feature to three linguistic labels (*Low*, *Medium* and *High*). Then, each real value F_j is replaced by its three values of membership to these fuzzy labels (μ_{jL}, μ_{jM} and μ_{jH}, respectively), and so, a new set of $3n$ features, $\boldsymbol{F'} = \mu_{1L}$, μ_{1M}, μ_{1H}, ..., μ_{nL}, μ_{nM}, μ_{nH} is constructed from the original set of features $\boldsymbol{F}$. The membership functions to linguistic labels are defined in a similar way to the form that has been used by [4].

Once defined the three membership functions for each feature F_j, a threshold value θ can be established (for example, 0.5) to discretize the original data in a binary way, according to any linguistic label from the defined labels *Low*, *Medium* and *High*. The discriminatory criterion for any label is simply defined by:

$$F'_{j\bullet} = \begin{cases} 1 & \text{if } \mu_{j\bullet}(x) \geqslant \Theta \\ 0 & \text{if } \mu_{j\bullet}(x) < \Theta \end{cases} \tag{1}$$

As is shown in Figure 2, for concrete values of threshold θ, specific zones of the feature domain for which none of the labels will be activated can exist (zone A in Figure 2). This fact must be interpreted as the specific value of the feature is not enough to assign it a significant linguistic label at the significance degree of membership fixed by threshold θ. On the other hand, one value can activate simultaneously two linguistic labels, since at the significance level given by θ, any assignment of the measure to a linguistic label is significant (zone B in Figure 2).

In this way, we have developed a method used to discretize numeric features into binary variables according to the definition of three linguistic labels, and therefore the method is defined in a fuzzy sets manner. Summarizing, given a data set $\boldsymbol{D}$ with m observations $\{x_1, ..., x_m\}$ about n numeric

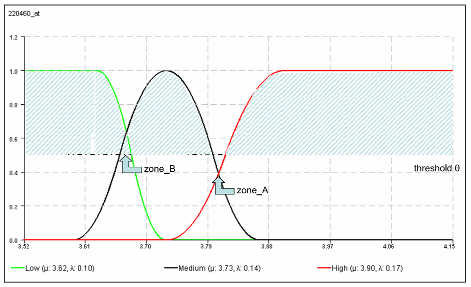

Figure 2: Example of membership functions for a given gene.

features $\boldsymbol{F} = \{F_1, ..., F_n\}$, namely, $x_i \in \mathbf{R}^n$, the fuzzy discretization process, defined above, transforms the original data set into another set with the same number of observations but a different number of features. The new data set ***D'*** has m observations which are now referred to as a set of $3n$ binary features, namely, $x'_i \in \{0,1\}^{3n}$. The real value of feature F_j for the observation x_i, denoted by x_{ij}, is replaced by the three binary values given by expression 1 for each linguistic label, that is to say, by the tuple $\langle F'_{jL}(x_{ij})$, $F'_{jM}(x_{ij})$, $F'_{jH}(x_{ij})\rangle$.

3.1.2 *Assembling a supervised fuzzy pattern of representative genes (FP)*

This section explains how to generate a fuzzy pattern from data, which is representative for a specific decision class. The process is carried out according to a supervised learning process from the available data as described below.

Given a subset of observations $D_i = \{x_{i_1}, x_{i_2}, ..., x_{i_m}\} \subseteq \boldsymbol{D}$, which have associated the same class label C_i, for any observation x_{i_1} ($i_1 \leq i_l \leq i_m$).

First, it is discretized with regard to the linguistic labels *Low*, *Medium* and *High* associated to each feature, F_j. Namely, the discrete values $F'_{jL}(x_{i_l j})$, $F'_{jM}(x_{i_l j})$, $F'_{jH}(x_{i_l j})$. are computed using the expression given by 1. Then, the three binary values for each feature are replaced by a single label, $F''_j(x_{i_l}) \in$ {L, LM, M, MH, H, *}. If only one of the three binary values is active, the respective label is assigned: L (*Low*), M (*Medium*), and H (*High*). As previously mentioned, a unique real value can activate simultaneously two linguistic labels, so it may occur that two binary values are activated - the possible cases are LM (*Low* and *Medium*) and MH (*Medium* and *High*). Finally, it is also possible that one value does not fire any linguistic label, and then, the label * is assigned. The assignment criteria for $F''_j(x_{i_l})$ is given completely by the following expression:

$$\begin{cases} \text{L} & \text{if } F'_{jL}(x_{i_l j}) = 1 \wedge F'_{jM}(x_{i_l j}) = 0 \wedge F'_{jH}(x_{i_l j}) = 0 \\ \text{LM} & \text{if } F'_{jL}(x_{i_l j}) = 1 \wedge F'_{jM}(x_{i_l j}) = 1 \wedge F'_{jH}(x_{i_l j}) = 0 \\ \text{M} & \text{if } F'_{jL}(x_{i_l j}) = 0 \wedge F'_{jM}(x_{i_l j}) = 1 \wedge F'_{jH}(x_{i_l j}) = 0 \\ \text{MH} & \text{if } F'_{jL}(x_{i_l j}) = 0 \wedge F'_{jM}(x_{i_l j}) = 1 \wedge F'_{jH}(x_{i_l j}) = 1 \\ \text{H} & \text{if } F'_{jL}(x_{i_l j}) = 0 \wedge F'_{jM}(x_{i_l j}) = 0 \wedge F'_{jH}(x_{i_l j}) = 1 \\ * & \text{if } F'_{jL}(x_{i_l j}) = 0 \wedge F'_{jM}(x_{i_l j}) = 0 \wedge F'_{jH}(x_{i_l j}) = 0 \end{cases} \tag{2}$$

Secondly, the fuzzy pattern (corresponding to the class C_i) is constructed from the discretized and summarized data, selecting those labels of features which are different to the label "*" and have an appearance relative frequency in set D_i equal to or greater than a predefined ratio Π ($0 < \Pi \leq 1$, for example, $\Pi = 2/3$). Formally, for each feature F_j, the appearance frequency of any label $E \in \boldsymbol{E} =$ {L, LM, M, MH, H, *} in the set D_i, $\Pi_{ij}(E)$, can be computed according to the expression given by:

$$\pi_{ij}(E) = \frac{\sum\limits_{i_1 \leqslant i_l \leqslant i_m} \delta_j(x_{i_l}, E)}{i_m} \tag{3}$$

where

$$\delta_j(x_{i_l}, E) = \begin{cases} 1 & \text{if } F''_j(x_{i_l}) = E \\ 0 & \text{otherwise} \end{cases}$$

Once, the frequency of each label is computed for every feature, a 3-tuple of the form $\langle$feature, label, frequency$\rangle$ is included in the fuzzy pattern of class C_i, only if its frequency exceeds the predefined ratio Π. Namely, the fuzzy pattern P_i is given by:

$$P_i = \left\{ \begin{array}{ll} \bigwedge\limits_{F''_j \in F''} & \left\langle F''_j, E^j, \pi^j \right\rangle : \\ & E^j = \arg\max\limits_{E \in E} \{\pi_{ij}(E)\} \wedge \\ & E^j \neq * \wedge \\ & \pi^j = \pi_{ij}(E^j) \geqslant \Pi \end{array} \right\} \tag{4}$$

The predefined ratio Π controls the degree of exigency for selecting a feature as a member of the pattern, since the higher the value of Π, the fewer number of features which make up the pattern.

The method presented here aims to construct a fuzzy pattern which is representative of a collection of observations belonging to the same decision class. The pattern's quality of fuzziness is given by the fact that the labels, which make it up, come from the linguistic labels defined during the discretization stage. On the other hand, if a specific label of one feature is very common in all the examples (belonging to the same class), this feature is selected to be included in the pattern and, therefore, a frequency-based criteria is used for selecting a feature as part of the pattern.

3.1.3 *Recognizing valuable genes (DFP)*

The goal of gene selection is to determine a reduced set of genes, which are useful to classify new samples given the existing knowledge. Now, we are interested in those genes that allow us to discriminate a given class with regard to the others. Here, we introduce the notion of *discriminant fuzzy pattern* with regard to a collection of FPs. A DFP version of a FP only includes those genes that can serve to differentiate it from the rest of the patterns. Therefore, the computed DFP for a specific FP is different depending on what other FPs are compared with it. It's not surprising that the genes used to discern a specific class from others (by mean of its DFP) will be different if the set of rival classes also changes. The pseudocode algorithm used to compute the final DFP containing the selected genes can be consulted on [4].

3.2 Wrapper methods

A *wrapper* method approaches the feature selection process by using an optimization algorithm that searches the space of possible subsets of attributes to find the best alternative. An objective function is defined that takes into account the accuracy of a classifier that is trained in the feature subset. This accuracy is measured by running an error estimation procedure on the classifier, typically a cross validation scheme. In addition to this accuracy measure, the objective function can also take into account other features of the solution that can be of interest. One example is the introduction of the number of genes in the objective function to reward solutions with a small number of features. As before, the aim of this system is to provide a framework that allows the implementation of a number of alternatives by considering distinct optimization algorithms.

In this work, *Evolutionary Algorithms (EAs)* [12] were developed to tackle this task. The proposed *EAs* use a set-based representation scheme where the chromosome is a set of integers that encode the indexes of the attributes (genes) used in the classifier.

A crossover operator is used, that is inspired on *uniform crossover* and works as follows: the genes that are present in both parent sets are kept in both offspring; the genes that are present in only one of the parents are sent to one of the offspring, selected randomly with equal probabilities. The mutation operator is a *random* mutation, that replaces a gene by a random value in the allowed range.

In this *EA*, variable-sized sets can be encoded and compete within the same population. Two additional mutation operators are defined in order to be able to create solutions with a distinct size:

- *Grow*: consists in the introduction of a new gene into the chromosome, whose value is randomly generated in the available range.
- *Shrink*: a randomly selected gene is removed from the genome.

All reproduction operators are used with equal probabilities to create new solutions. The operators are implemented taking into consideration the need to comply to the constraints imposed by the minimum and maximum set size and also to avoid repeated elements in the sets. In the experiments reported in this work, the minimum size is always set to 1.

It is important to mention that both the proposed representation is mathematically equivalent to a binary representation, since there is a one-to-one correspondence between the solutions. This means that the underlying search space is the same. However, the reproduction operator and initialization schemes create differences in the way the search space is explored, that can result in distinct outcomes in the end of the optimization process. The reason to use this representation is the fact that it is more compact and that allows a better control of the size of the encoded sets, using the aforementioned operators.

The selection procedure consists in converting the fitness value into a linear ranking in the population, and then applying a roulette wheel scheme. In each generation, 50% of the individuals are kept from the previous generation, and 50% are bred by the application of the reproduction operators. An elitism value of 1 is used, allowing the best individual of the population to be always kept.

An initial population is randomly created, taking into account the fact that the sets do not allow repetitions. The size of the individuals is randomly generated in the range $[L, U]$, where U is also defined as the maximum size of the individuals during evolution. The termination criterion is based on a fixed number of generations G.

3.3 Classifiers

The flexibility of the framework was achieved, in the case of the classification components, by making use of the *WEKA* open source data mining software [17]. Indeed, the system allows the user to choose any of the classifiers and algorithms that are available in *WEKA*, providing a wide range of alternatives that include, among others, *decision trees*, *classification rules*, *neural networks*, *support vector machines* or *instance-based learning*.

In the experiments conducted during this work, the following classifiers/ algorithms were used:

- *J48* – a classification decision tree based on the C4.5 algorithm;
- *IB1* – a *1-Nearest Neighbor*;
- *SMO* – a *Support Vector Machine*, using the *Sequential Minimal Optimization* training algorithm.

The use of the *WEKA* software is not confined to the classifiers, but is also useful in the definition of error estimation methods. The proposed system allows the user to select which method is used to estimate the error, e.g., *k-fold cross-validation*, *holdout* or *leave one out*.

4. EXPERIMENTS

4.1 Datasets

The *Lung* dataset [2] consists of 254 human microarray samples with 12625 genes. There are 5 distinct classes, 186 of these are adenocarcinoma samples, 20 are pulmonary carcinoids, 17 normal lung specimens, 21 squamous cell lung carcinomas and 6 small-cell lung carcinomas.

Acute Myeloid Leukemia (AML) is not a single disease but a group of neoplasms with diverse genetic abnormalities and variable responses to treatment [16]. The *New England* dataset used is a 105 samples subset of the original dataset described in the afore cited article. It consists of 4 classes and 22283 genes. Blasts and mononuclear cells were purified from bone marrow or peripheral blood aspirates of acute myeloid patients. Samples contained 80-100 percent blast cells after thawing, regardless of the blast count at diagnosis. Patients were classified into 4 subgroups: (i) 19 APL, (ii) 64 M5, (iii) 14 AML with inv(16) plus a group of 8 healthy donors.

4.2 Methodology and implementation

The system was implemented in the *Java* programming language using *AIBench* (http://www.aibench.org), a MVC-based Java application framework that eases the connection, execution and integration of operations with well defined input/output. The code of the application is freely available in the AIBench web site.

Regarding the *EA*'s setup, the population size was set to 100 individuals and the process is stopped after 200 generations. The fitness function was computed by using the accuracy of a 5-fold cross-validation scheme.

The proposed approach was tested in the two datasets, with the three classifiers identified in the previous section.

Two variants were considered for the upper bound to the number of genes(U): 20 and 50 genes. Thus, each test combination is one dataset, one maximum number of genes and one algorithm.

Due to CPU time constraints, a *5 times 10-fold cross-validation* over the all dataset was performed for each combination. In each iteration of this process, the test fold is not used in any way by the classifier nor the *EA* that performs feature selection.

4.3 Results

The first step in each run is the pre-processing stage, i.e. the application of the *DFP* filter method. After this process, the datasets are reduced to a mean of 796 and 402 genes, for the *Lung* and *New England* datasets respectively.

The results of the *EA* are presented in Tables 1 and 2, in terms of the accuracy of the final classifier obtained in each run. The tables show the classifier, the maximum number of genes allowed, the number of genes selected by the *EA*, the validation accuracy (obtained by the cross-validation used in fitness evaluation) and the test accuracy of the final classifier. The last three are shown in terms of the mean and 95% confidence intervals. In Figures 3 and 4, two boxplots with the test accuracies grouped by classifier and maximum number of genes are shown, for both datasets.

Classifier	Max Genes	Genes	Validation Accuracy	Test Accuracy
IB1	20	15.22 ± 0.67	99.71 ± 0.18	94.11 ± 1.14
J48	20	11.61 ± 1.63	97.58 ± 0.33	87.9 ± 1.35
SMO	20	16.2 ± 0.67	99.29 ± 0.33	94.29 ± 0.76
IB1	50	33.39 ± 3.92	99.67 ± 0.15	95.3 ± 1.04
J48	50	29.22 ± 3.41	97.52 ± 0.24	88.89 ± 1.61
SMO	50	34.62 ± 4.28	99.63 ± 0.37	94.35 ± 1.66

Table 1: Results obtained by the *EA* on the *Lung* dataset.

Classifier	Max Genes	Genes	Validation Accuracy	Test Accuracy
IB1	20	12.61 ± 1.15	98.89 ± 0.46	87.14 ± 2.68
J48	20	10.33 ± 1.58	95.87 ± 0.54	78.89 ± 2.39
SMO	20	12.94 ± 0.86	99.05 ± 0.49	88.1 ± 3.17
IB1	50	27.22 ± 2.45	98.49 ± 0.57	91.43 ± 3.08
J48	50	24.17 ± 3.19	95.24 ± 0.64	77.94 ± 2.11
SMO	50	26.83 ± 2.92	98.17 ± 0.59	92.06 ± 2.26

Table 2: Results obtained by the *EA* on the *New England* dataset.

Examining the results, it is clear that good results (above 90% accuracy) can be obtained using a small number of genes (around 15 in the *Lung* and 27 in the *New England* dataset). This shows the good performance of the *EA* in the process of gene selection. It is interesting to notice that the results obtained on the cross validation used in fitness evaluation (a value reported by some authors as the final performance measure) is very high (typically around 98% or 99% in several cases).

A comparison of the results obtained by the classifiers shows that *J48* seems inadequate to this task, showing some problems in generalization (the difference between validation

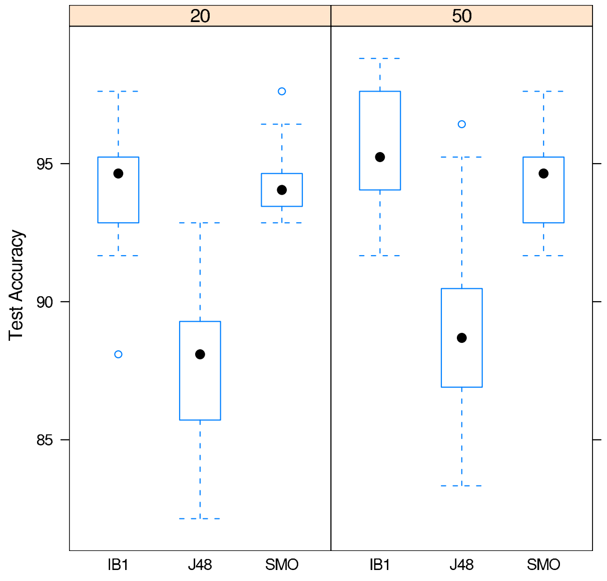

Figure 3: Boxplot for the Lung dataset.

and test accuracies is very large). The *IB1* and *SVM* are in similar levels of accuracy and it is not possible to prove any statistical difference between the two. The increase in the number of genes is, in most cases, reward by a small improvement in the results, although this is not significant in some cases and there is even a decrease in two of the setups.

5. CONCLUSIONS AND FURTHER WORK

In this work, a flexible computational framework was proposed to the task of gene selection in classification of DNA microarray data. The approach is to consider an optimization engine based on an *Evolutionary Algorithm*, that encodes solutions using a set-based representation with variable sized chromosomes. This *EA* works as a *wrapper* layer over a classifier, evaluating each solution (subset of genes) by considering its estimated accuracy.

The proposed system is quite flexible allowing the use of a number of distinct classifiers, by interacting with the *WEKA* software and making use of all the alternatives it provides. Furthermore, a number of filter methods can be used in the pre-processing stage and the fitness function can be molded to integrate other features (e.g. the number of genes).

The work reported here is still ongoing, and a number of new features are expected to be integrated, such as new filter methods and the possibility of defining classifiers that are not included in *WEKA*. Additionally, more tests on different datasets will be conducted in the near future to validate the approach.

Acknowledgments

The authors wish to thank the Portuguese Foundation for Science and Technology for their support through project ref. POSC/EIA/59899/2004, partially funded by FEDER.

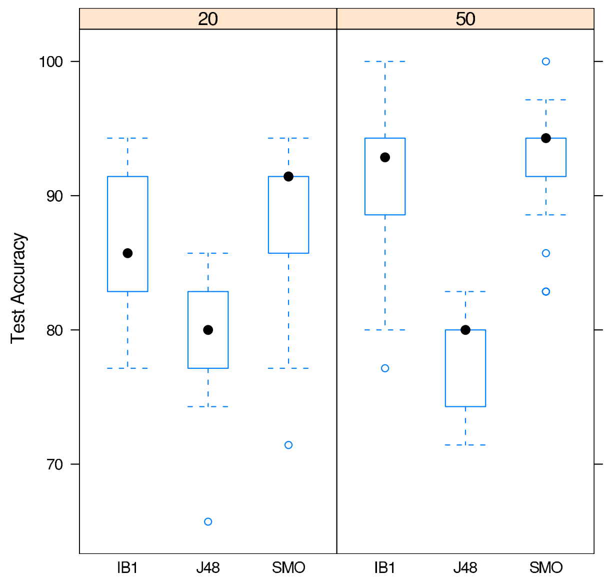

Figure 4: Boxplot for the New England dataset.

6. REFERENCES

[1] A. Ben-Dor, N. Friedman, and Z. Yakhini. Scoring genes for relevance. Technical Report AGL-2000-13, Agilent Laboratories, 2000.

[2] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenoma subclasses. *PNAS*, 98:13790–5, 2001.

[3] J. M. Deutsch. Evolutionary algorithms for finding optimal gene sets in microarray prediction. *Bioinformatics*, 19(1):45–52, 2003.

[4] F. Diaz, F. Fdez-Riverola, and J. M. Corchado. Gene-cbr: a case-based reasoning tool for cancer diagnosis using microarray datasets. *Computational Intelligence*, 22(3/4), 2006.

[5] H. Fröhlich, O. Chapelle, and B. Schölkopf. Feature selection for support vector machines by means of genetic algorithms. In *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 142, Washington, DC, USA, 2003. IEEE Computer Society.

[6] B. Hanczar, M. Courtine, A. Benis, C. Hennegar, K. Clment, and J. Zucker. Improving classification of microarray data using prototype-based feature selection. *ACM SIGKDD Explorations Newsletter*, 5(2):23–30, 2003.

[7] J. Jaeger, R. Sengupta, and W. Ruzzo. Improved gene selection for classification of microarrays. In *Proceedings of the Pacific Symposium on Biocomputing (PSB2003)*, pages 53–64, 2003.

[8] T. Jirapech-Umpai and S. Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6:148, 2005.

[9] E. Keedwell and A. Narayanan. Genetic algorithms for gene expression analysis. In *EvoWorkshops*, pages 76–86, 2003.

[10] L. L., D. T.A., W. C.R., L. A.J., and P. L.G. Gene assessment and sample classification for gene expression data using a genetic algorithm / k-nearest neighbor method. *Combinatorial Chemistry & High Throughput Screening*, 4(8):727–739, 2001.

[11] J. J. Liu, G. Cutler, W. Li, Z. Pan, S. Peng, T. Hoey, L. Chen, and X. B. Ling. Multiclass cancer classification and biomarker discovery using ga-based algorithms. *Bioinformatics*, 21(11):2691–2697, 2005.

[12] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, USA, third edition, 1996.

[13] C. H. Ooi and P. Tan. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics*, 19(1):37–44, 2003.

[14] S. Peng, Q. Xu, X. B. Ling, X. Peng, W. Du, and L. Chen. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Lett*, 555(2):358–362, December 2003.

[15] H. Qi. Feature selection and knn fusion in molecular classification of multiple tumor types. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Science (METMBS2002)*, 2002.

[16] P. J. Valk, R. G. Verhaak, M. A. Beijen, C. A. Erpelinck, S. Barjesteh van Waalwijk van Doorn-Khosrovani, J. M. Boer, H. B. Beverloo, M. J. Moorhouse, P. J. van der Spek, B. Löwenberg, and R. Delwel. Prognostically useful gene-expression profiles in acute myeloid leukemia. *N Engl J Med*, 350(16):1617–1628, April 2004.

[17] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.

[18] E. Xing, M. Jordan, and R. Karp. Feature selection for highdimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML2001)*. Morgan Kaufmann, 2001.