A parallel Genetic Programming for Single Class Classification

Cuong To
Institute of System and Synthetic Biology,
University of Évry
5 rue Henri Desbruères
91030 Evry Cedex, France
+33 (0)1 69 47 44 43
cuong.to@issb.genopole.fr

Mohamed Elati
Institute of System and Synthetic Biology,
University of Évry
5 rue Henri Desbruères
91030 Evry Cedex, France
+33 (0)1 69 47 44 43
mohamed.elati@issb.genopole.fr

ABSTRACT
In this paper, we present an algorithm based on genetic programming for single (one) class classification that uses one set containing similar patterns in training process. This type of problem is called single (one) class classification, a novel detection. The proposed algorithm was tested and compared to seven other traditional methods based on two publicly available transcriptomic and proteomic time series datasets and two public breast cancer datasets. The results show that the algorithm could find most similar patterns in the databases with rather low misclassification rates. We also applied parallel genetic programming for this algorithm and it proves that the island model can give better solutions than sequential genetic programming.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving – *Heuristic methods.*

General Terms
Algorithms.

Keywords
Single class classification, one class classification, genetic programming, island model.

1. INTRODUCTION
In the field of pattern classification, this problem is normally divided into unsupervised learning (cluster methods) and supervised methods. The supervised methods are based on expert knowledge that are represented as training sets. Supervised methods can be classified into binary classification, multi-class classification (multi-class classification is usually converted into multi binary classifications), and single class classification. The training set of binary classification consists of two sub sets, namely positive set and negative set. The positive set contains similar (target) patterns which we would like to search; and the negative set is opposite of the positive set. The negative set is very important because it helps classifiers to recognize patterns which differ from the positive set. However there are some features of the negative set that can affect searching results such as type of patterns and number of patterns.

Moreover, a fundamental assumption of binary classification is that the training and test set have identical distribution [23] but this assumption may not hold in practice. Li et al. [23] studied a particular problem where the positive set is identically distributed but the negative set may or may not be so and he gave a significant conclusion that the negative set should not be used in this setting and learning from positive and unlabeled sets fits this special case quite well.

In recent years, the single (one) class classification has rapidly emerged in pattern recognition field; the motivation is that the negative set is either not present or not properly sampled. According to Khan and Madden [1], single class classification can be classified under three types 1) learning with positive examples only; 2) learning with positive examples and some amount of poorly distributed negative examples; 3) learning with positive and unlabeled data.

Curry and Heywood [26, 27] proposed multiobjective one class genetic programming, dynamic page-based linear genetic programming was fulfilled as underlying learner.

Tax and Duin [3], [4] introduced a method called support vector data description (SVDD), inspired by support vector machine. This method computes a sphere with minimal volume covering a set of patterns. The sphere boundary is described by support vectors. Furthermore, the hyper-sphere model of SVDD can be made more flexible by introducing kernel functions. Tax [2] exerted a polynomial and a Gaussian kernel and found that the Gaussian kernel gives better results for most data sets.

Scholkopf and Smola [5] constructed a hyper-plane which is maximally distant from origin, with all data points lying on the opposite side from the origin and such that the margin is positive.

Yu [7] proposed a single class classification algorithm called mapping convergence that computes an accurate boundary of the target class from positive and unlabeled data, and without labeled negative data. The author concluded that without the negative set, single class support vector machines requires a much larger amount of positive training data to induce an accurate class boundary.

To and Vohradsky [6] considered the single class classification as the nonlinear programming. This algorithm finds the hyper-plane that minimizes the distances from the hyper-plane and all points of the positive set. The authors suggested using genetic algorithm to solve the nonlinear programming and the parallel model of

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

genetic algorithm called island model was also used to improve the results. The algorithm was applied to microarray data.

Gaussian process methods are widely applied for regression and classification problems. Gaussian process defines a distribution over functions based on the central assumption that all function values are jointly normal distributed. Kemmler et al. [24] analyzed different measures derived from Gaussian process in single class classification for visual object recognition.

Peng et al. [20] presented a threefold classifier from positive and unlabeled documents. First, reliable negative documents are identified by improved 1-DNF algorithm. Second, a set of classifiers are built using support vector machine. Finally, genetic algorithm search weights of classifiers to build the final classifier.

A naïve Bayes algorithm [21] learns from positive and unlabeled documents. The results show that performance of this algorithm can be comparable with naïve Bayes which learns from labeled data.

De Comite et al. [8] proposed an algorithm based decision tree which can take three sets, namely labeled examples, positive examples, and unlabeled data as input. The decision tree is generated by a modified version of C4.5. The proposed algorithm was tested based on two public data sets called kr-vs-kp and adult.

Learning with positive and unlabeled examples [22] was converted into learning with noise by labeling all unlabeled examples as negative and use a linear function to learn. In order to minimize the expected sum of false positive and false negative error frequencies, the real-value conditional probability of observing a positive label by performing logistic regression was learnt.

Many various single class classification algorithms using global Gaussian approximation, 1-nearest neighbor, neural network, statistic methods were mentioned in [9]-[11].

In recent years, parallel models have been applied in evolutionary computation and they show not only increase speed, but also give high quality solutions [6]. In this paper, the island model was applied and the results testify that the island model outperforms sequential genetic programming.

In this work, we would like to introduce an algorithm using a set of similar patterns for the training process. This kind of methods is called single (one) class classification [1], [5]. The proposed algorithm that is based on genetic programming is to search a polynomial function describing the similarity among patterns of the training set.

The outline of the paper is as follows. The section 2 introduces some basic definitions, the problem statement, and the algorithm. In section 3, we describe how our approach is evaluated and compared to state of the art classification algorithms on a number of real datasets, obtaining very good performance. Finally, we conclude with a brief discussion.

2. METHOD

Let TS = $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ be the training set, where $\mathbf{p}_i \in \Re^n$ is a pattern (point) and m is number of pattern in the training set. In this paper, the term 'pattern' or 'point' is a time series vector $\mathbf{p}_i = (p_{i1}, p_{i2}, ..., p_{in})^{\mathrm{T}}$, where n is number of time points (measurements). At the beginning, the training set, TS, consists of a small number of similar patterns. During the training process, the genetic programming searches a curve which can fit all patterns of the training set. Then patterns of database, which are close to the curve, are selected. The selected patterns are desired to have similarity to the template pattern.

Average Euclidean distance between two patterns $\mathbf{p}_r = (p_{r1}, p_{r2}, ..., p_{rn})^{\mathrm{T}}$ and $\mathbf{p}_s = (p_{s1}, p_{s2}, ..., p_{sn})^{\mathrm{T}}$ is given by:

$$Dis(\mathbf{p}_r, \mathbf{p}_s) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_{ri} - p_{si})^2} \quad (1)$$

Because each element of pattern is within the range [0, 1], so $Dis(\mathbf{p}_r, \mathbf{p}_s)$ is within [0, 1]. The value of zero means that $\mathbf{p}_s$ and $\mathbf{p}_r$ are identical. The arithmetic mean of the distances between the pattern, $\mathbf{p}_r$, and other patterns of the training set is given by

$$M(\mathbf{p}_r) = \frac{\sum_{i=1}^{m} Dis(\mathbf{p}_r, \mathbf{p}_i)}{m} \quad (2)$$

The pairs of values $[\mathbf{p}_i, M(\mathbf{p}_i)]$ ($i = 1..m$) constitute the regression problem. There are many methods to solve the regression problem. According to Koza [12], genetic programming (GP) is rather powerful method. Therefore, we decided to use GP to search for a function that satisfies the following criterion.

$$f(\mathbf{p}_r) \approx M(\mathbf{p}_r) \quad (3)$$

In fact, if there exists the function describing (3), the type of that function will not be known in advance. Of course, we can use different mathematical methods to approximate functional relationship; in this algorithm we use the polynomial function because the polynomial function can be used to approximate to any kinds of function, in principle. Therefore the fitness function is given by

$$\text{Fitness} = \sum_{i=1}^{m} \left| f(\mathbf{p}_i) - M(\mathbf{p}_i) \right| \quad (4)$$

where

- m: number of patterns of the training set.
- $f(\mathbf{p}_r)$: polynomial function.
- $M(\mathbf{p}_i)$: mean of the distances between pattern, $\mathbf{p}_i$, and other patterns of the training set computed by (2).

2.1 Classification

After the function is determined by genetic programming, the function values of all patterns of the training set are calculated. Based on these values, the mean, μ and standard deviation, σ can be extracted. If the function value, $f(\mathbf{a})$ of the tested pattern, $\mathbf{a}$ is within the range $[\mu - 2\sigma, \mu + 2\sigma]$, then the tested pattern, $\mathbf{a}$ is defined as similar to the training set.

We selected the above criterion because most of natural phenomenon comply the normal distribution and 95% of cases fall into the range $[\mu - 2\sigma, \mu + 2\sigma]$. Although any criterion which analyzes distribution of function values of all patterns of the training set can be fulfilled, we selected the above criterion because it is quite simple and results are satisfied.

2.2 Performance Measurements

Performance of the algorithm was evaluated by using two popular indicators [6], namely sensitivity (*Se*) and specificity (*Sp*), defined as follows

$$Se = \frac{TP}{|C|} \tag{5}$$

$$Sp = \frac{|R| - FP}{|R|} \tag{6}$$

with

- *TP (true positive)*: the classifier predicts that the pattern is in the training set and the pattern belongs to the training set.
- *FP (false positive*): the classifier predicts that the pattern is in the training set but the pattern does not belong to the training set.
- |C|: total number of patterns which are similar to the patterns of the initial training set.
- |R|: total number of patterns which are different from the patterns of the training set.

The sensitivity means that the rate of patterns is correctly classified. The value one means all similar patterns are found. The specificity is the rate of misclassified patterns. The value one means no misclassified patterns found. The values of both *Se* and *Sp* should approach one for good performance.

2.3 Genetic Programming Parameters

The influence of the genetic programming parameters, i.e. population size, probability of crossover, and number of generations, on the performance of the algorithm was tested with a pattern of an arbitrarily chosen size $n = 21$ (the size is typical for the temporal proteomic or transcriptomic experiments). As the effect of the parameters was seen only for the level of added noise greater than or equal 30%, we have created a test set by adding 30% noise to the patterns. The influence of the parameters was evaluated using standard ROC curves.

Population size (number of trees in the population) is one of the most important parameters of the GP. If the population size is too small, it is difficult to find good solution; and if it is too large, the computing time is long and the algorithm tends to overfit. Results of *Se* and *Sp* for different values of GP parameters are summarized in Figures 1–3.

Figures 1–3 show a very good overall performance of the algorithm when both *Sp* and *Se* are close to 1 for most of the parameter combinations. The algorithm achieves the best performance for a population size equal to 1000 ($Se = 1$, $Sp = 0.89$) where the maximum number of generations does not influence *Se* until the value of 700 when the selectivity decreases, and when *Sp* monotonously increases with the number of generations used. The best combination of *Sp* and *Se* is reached for 500 and/or 700 generations. The sensitivity *Se* increases with the probability of crossover, when for the value of 0.8 the $Se = 1$, *Sp* fluctuates around 0.9. The best combination of *Se* and *Sp* is reached when the probability of crossover is equal to 0.9. The maximum depth of the initial random tree was arbitrarily chosen as 7.

The selected control parameters of genetic programming are listed in Table 1.

2.4 Parallel Genetic Programming

Among four major types of parallel genetic programming [12], the island model is rather complicated and is the most popular type [14]. In this section, the island model is generally introduced. In the island model, the population is partitioned into sub populations. Each sub population which is assigned to one processor and runs independently is called island. After a predefined number of generations, islands exchange trees with each other called migration. This model has been being applied for many problems [15–17] and shows that it not only increases performance of algorithms but also gives results better than sequential genetic programming.

In order to use the island model, we have to determine some parameters as: topology, migration rate, migration frequency, and sub population size.

There are some topologies as grid, ring, and random one. Fernandez de Vega [17] introduced a random topology and compared it with grid and ring topology. He also gave the significant conclusion that if all other parameters are fixed, there are no significant differences when topologies are changed.

By testing on four problems (2 classic and 2 real-life problems: even parity 5, ant problem, routing and placing circuits on FPGAs, and medical diagnosing), Fernandez de Vega [17] showed that the best migration rate is between 5% and 10%. He also did a wider study on comparing different migration frequencies and gave the conclusion that the best convergence results appear when about 10% of trees from each sub population are sent every 5–10 generations.

Concerning on sub population size, Calegari [16] showed that the solution which obtained with 4 islands of 40 trees each is better than that found with a single population of 160 trees. The solution obtained with 40 islands of 4 trees each is even better.

According to the results above, we applied parallel genetic programming for the algorithm with the following parameters: topology is ring, migration rate is from 5% to 10%, migration is executed every 10 generations, and sub population size is 500 and 260 for 2 islands and 4 islands, respectively. And the obtained results (Tables 4, 5, 8 and 9) show that the above parameters of the island model are rather suitable for this algorithm.

3. EXPERIMENTS

In order to evaluate the performance of the proposed algorithm, two real biological data namely response of fibroblasts to serum and Caulobacter and two breast cancer data were exerted. The comparisons between the proposed algorithm (Sequential GP) and seven other methods, namely binary support vector machine (Binary SVM) [5], single class support vector machine (Single SVM) [5], LogitBoost, logistic regression (LR) [25], linear discriminant analysis (LDA) [25], linear least square regression (LS) [25], and parallel genetic algorithm (Single GA) [6] were also fulfilled.

The training processes of binary classification methods need the positive and the negative sets; whereas the proposed algorithm uses one set for the training process. So the training set of the proposed algorithm is the positive set of binary classification methods and the negative set of binary classification methods is randomly selected.

3.1 Transcriptomic and Proteomic Data

The two data sets previously analyzed using clustering methods were employed. We classified the *C. crescentus* cell cycle controlled protein profiles [28–29] obtained from SWICZ server (http://proteom.biomed.cas.cz) and 517 genes monitored in 19 different time points using DNA chips, representing the response of fibroblasts to serum [30] (http://genome-www.stanford.edu/serum/clusters.html). Cluster analysis identified groups of profiles according to predefined similarity metrics by building a hierarchical tree of similarity between individual profiles and clusters of profiles. Cluster analysis partitioned the dataset into a set of disjoint groups according to a defined similarity threshold. As the method excludes user interaction, the groups of patterns were identified after the analysis of the clustering tree. We have chosen a different approach that consists of defining a template and performing the search of desired profiles using different algorithms – sequential GP, parallel GP and the published algorithms mentioned above.

In order to make possible the mutual comparison of these methods and the cluster analysis, we selected the training sets for GP and the other algorithms by random selection from the clusters identified by the cluster analysis. Therefore, for each of the clusters, an initial training set consisting of randomly selected patterns from each cluster was created and the GP was applied for identification of other members of the cluster. The true and false positives were derived from a comparison with the results of the cluster analysis.

Average expression profiles for the proteomic set (*C. crescentus*), and the clusters of transcriptomic set (fibroblast) identified by the original cluster analysis are shown in Figures 4–6 and the results for seven different algorithms are summarized in Tables 6 and 7.

3.2 Breast Cancer Data

The two breast cancer data that were normally used in many previous classification methods are also introduced. The Wisconsin Breast Cancer Database [18] was obtained from the University of Wisconsin Hospitals, Madison. Each instance has one of 2 possible classes: benign or malignant tumor. There are 458 instances for benign class and 241 instances for malignant class. The Wisconsin Diagnostic Breast Cancer was first used in [19]. There are 569 instances each of which belongs to benign class or malignant class (357 benign, 212 malignant). Each instance is described by 30 real-valued attributes. Attributes are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

For both of breast cancer data sets, the training set and the test set were randomly selected and the rate of them is 50% – 50%. The results are listed in Tables 2 and 3.

The Tables 4, 5, 8 and 9 show the results of sequential and parallel genetic programming (Parallel GP). For parallel genetic programming, the island model was applied.

3.3 Discussion

Clusters from microarray and proteomic experiments were chosen to cover different sizes of the target group and different group correlations, which range from 0.7 to 0.3. Results are summarized in Tables 6 and 7. As a measure of accuracy, we allowed the comparison of the different algorithms, and the original results of hierarchical clustering were chosen. Tables 6 and 7 show that the desired sensitivity was close to 1 for most of the algorithms. Variations can be seen between the transcriptomic and proteomic data which reveals that most of the non GP algorithms failed on the transcriptomic dataset. The GP methods showed the best results, with *Se* always equal to 1 and $Sp \sim 0.9$.

Higher false positive rates, reflected in *Sp*, mean that the algorithm found expression profiles which did not belong to the cluster, defined by the hierarchical clustering algorithm. The reason is, that the hierarchical clustering groups objects according to their mutual similarity starting from the most similar pair and continuing by adding new groups or individuals, forming a clustering tree. Therefore, the similarity within the cluster is given not only by the similarity among the profiles, but also by the similarity to other closely related objects or clusters. For well separated clusters, this does not bring about any problems and the clusters are correctly identified; but for more fuzzy overlapping clusters, the classification can fail. In such a case, the single class classification based on pattern recognition gives better results than the clustering. The pattern recognition methods were able to find all similar profiles which were otherwise assigned by the clustering algorithm to different clusters. In this case, the GP and the other algorithms followed similar trends.

Comparison of GP with SVM showed better performance of the GP algorithm, especially, when the parallel computational scheme was employed. Tables 4, 5, 8 and 9 show that in the parallel processing scheme, the accuracy of the algorithm increases with a growing number of islands. The advantage of GP is also in the requirement of only one training set whereas SVM requires two sets – the first comprised of profiles belonging to the target group, the second comprised of the rest.

Another advantage of the presented GP scheme is the possibility to include a user interaction to the training process. After each training loop, the user can (but does not have to) check the results, change the training set and restart learning of the algorithm with this new set. This scheme allows for high flexibility in the definition of within group pattern similarity.

In genetic programming, the influence of the individual variables and their combination in the best tree on the result can be done by analyzing the tree. In this case, this information is not necessary as we are not interested in the particular influence of the variables, but instead, the goal is to identify the profiles. If the program satisfies this criterion, the goal is reached. GP is a random process where the final tree is the result of random steps of crossover and reproduction and it can not be guaranteed that the final tree is optimal. This problem is usually bypassed by running the learning procedure several times and selecting the best result. Our procedure allows repeated training but in order to save computation time, it has not been used.

4. CONCLUSIONS

The presented algorithm falls into a class of single (one) class classification which has rapidly emerged in pattern recognition in the last few years. For the single class classification, we want, in a given dataset, to estimate a subset such that the probability that a test point drawn from the dataset lies outside of the subset equals some a priori specified value between 0 and 1. The goal is to find a function which is positive for the desired subset and zero or negative for the complement. In this paper, we presented evidence that genetic programming is suitable for this task, allowing for the

identification of user defined gene expression time series templates in a large set of profiles.

The demand for identification of user defined templates of gene expression profiles increases with the availability of large scale gene expression data when a microarray or proteomic experiment covers whole cell cycles or other time evolving processes. Typical genome size and thus the number of genes immobilized on a microarray exceed tens of thousands. Thus the number of time series of an experiment also exceeds this number. To search through such a database is a nontrivial task. With the increasing knowledge about the regulation of gene expression, such datasets can be approached with already existing knowledge of the system. Therefore the initial classification of the profiles into disjoint clusters can now be replaced by a targeted search for genes which have a profile similar to the profile of a gene with already known function. Such genes are either under control of the same promoter or they participate in the same regulatory process. Their identification is essential for elucidating of their control and their role in the studied process. In such a case, approaching the problem as a single class problem is appropriate. Here, we showed that the parallel genetic programming gives very good results and in all tested cases outperformed previously published algorithms.

The disadvantage of the evolutionary methods in general is their high computation intensive. We bypassed this problem by introduction of a parallel computational scheme which greatly increases the speed of computation. Moreover, the parallel scheme is suggested here improves the performance of the presented algorithm. Nowadays the multiprocessor machines are readily available and the parallel programming is no longer a domain of large computers. Therefore, implementation of the parallel algorithm presented here is feasible.

Our algorithm combines robustness of genetic programming and the speed of parallel computing with the desired flexibility given by the user interaction. Once trained, the program can be applied to any other database of the same kind. Therefore it is possible to create a repository of classifiers for different template types and use them for different databases or apply them repeatedly to a growing database of proteomic or transcriptomic expression profiles.

Table 1. Genetic programming parameters (powX is power of X)

Population size	1000
Maximum generation	500
Probability of crossover	0.90
Probability of reproduction	0.10
Terminal set	$p_1, p_2, \ldots, p_n$
Function set	+, -, ×, pow2, …, pow10

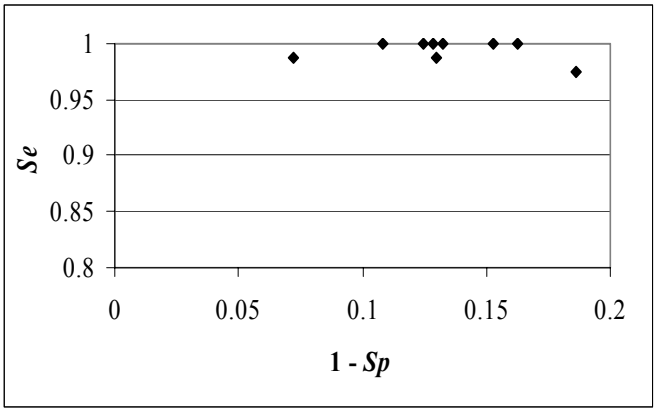

Figure 1. ROC analysis of population size.

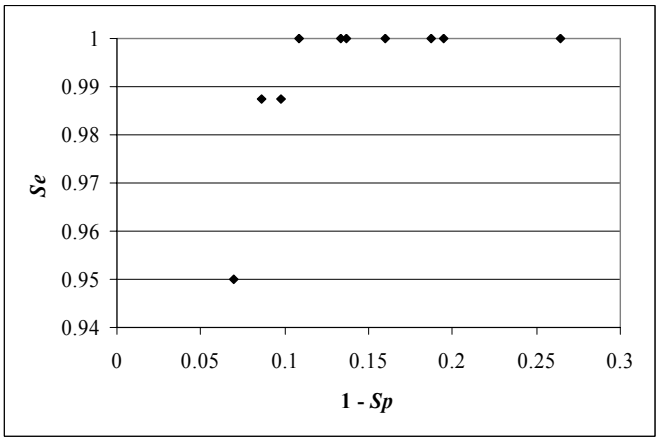

Figure 2. ROC analysis of maximum generations.

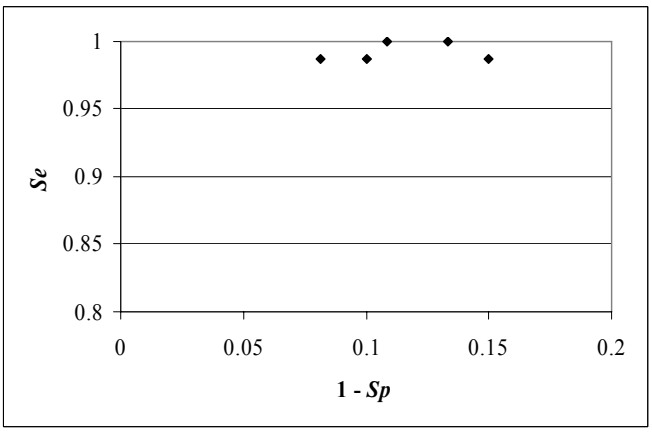

Figure 3. ROC analysis of probability of crossover.

Table 2. Wisconsin breast cancer database results of eight algorithms

Class	Sequential GP		Binary SVM		Single SVM		LogitBoost		LR		LDA		LS		Single GA	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*
Normal	0.996	0.791	0.959	0.753	1.000	0.347	1.000	0.381	0.991	0.389	0.973	0.795	1.000	0.360	1.000	0.803
Malignance	1.000	0.919	0.983	0.939	0.983	0.908	1.000	0.858	1.000	0.858	0.975	0.869	0.992	0.802	1.000	0.926

Table 3. Wisconsin diagnostic breast cancer results of eight algorithms

Class	Sequential GP		Binary SVM		Single SVM		LogitBoost		LR		LDA		LS		Single GA	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*
Normal	0.961	0.816	0.983	0.717	0.511	0.014	1.000	0.594	1.000	0.519	1.000	0.561	1.000	0.439	1.000	0.830
Malignance	0.906	0.765	0.877	0.966	0.953	0.711	0.943	0.675	0.943	0.703	0.981	0.557	1.000	0.090	0.972	0.812

Table 4. Wisconsin breast cancer database results of parallel genetic programming

Class	Sequential GP		Parallel GP (2 islands)		Parallel GP (4 islands)	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*
Normal	0.996	0.791	0.996	0.836	0.996	0.862
Malignanc e	1.000	0.919	1.000	0.925	1.000	0.940

Table 5. Wisconsin diagnostic breast cancer results of parallel genetic programming

Class	Sequential GP		Parallel GP (2 islands)		Parallel GP (4 islands)	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*
Normal	0.961	0.816	0.970	0.833	0.970	0.854
Malignanc e	0.906	0.765	0.956	0.801	0.956	0.833

Table 6. Transcriptomic database results of seven algorithms (null value means algorithm does not work)

Cluster	Sequent GP		Binary SVM		Single SVM		LogitBoost		LR		LDA		LS	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*
263 – 296	1.000	0.868	0.900	0.830	1.000	0.841	0.600	0.671	0.500	0.729	0.650	0.750	0.650	0.795
301 – 343	1.000	0.871	0.954	0.886	1.000	0.850	1.000	0.719	1.000	0.751	1.000	0.740	1.000	0.740
394 – 407	1.000	0.964	1.000	0.759	0.857	0.958	-	-	0.714	0.525	1.000	0.698	0.143	0.525
493 – 517	1.000	0.947	1.000	0.782	1.000	0.878	1.000	0.439	0.800	0.376	1.000	0.411	1.000	0.411

Table 7. Proteomic database results of seven algorithms

Cluster	Sequent GP		Binary SVM		Single SVM		LogitBoost		LR		LDA		LS	
	Se	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*	*Se*	*Sp*
1-3	1.000	0.909	1.000	0.719	0.542	0.959	1.000	0.744	1.000	0.702	1.000	0.653	1.000	0.636
4-7	1.000	0.922	1.000	0.861	0.800	0.809	1.000	0.930	1.000	0.887	1.000	0.878	1.000	0.844
10-14	1.000	0.944	0.973	0.917	0.946	0.870	1.000	0.741	1.000	0.750	1.000	0.713	1.000	0.713
17-19	1.000	0.718	1.000	0.637	0.800	0.889	1.000	0.748	1.000	0.748	0.900	0.741	0.700	0.756
20-23	1.000	0.965	1.000	0.948	0.767	0.844	1.000	0.635	1.000	0.635	1.000	0.626	1.000	0.626
1-7	1.000	0.901	0.963	1.000	0.889	0.989	1.000	1.000	1.000	1.000	1.000	0.967	1.000	0.967
17-23	0.900	0.714	0.975	0.924	0.800	0.857	0.975	0.848	0.975	0.838	1.000	0.924	1.000	0.924

Table 8. Transcriptomic database results of parallel genetic programming

Cluster	Sequent GP		Parallel GP (2 islands)		Parallel GP (4 islands)	
	Se	***Sp***	***Se***	***Sp***	***Se***	***Sp***
263 – 296	1.000	0.868	1.000	0.901	1.000	0.905
301 – 343	1.000	0.871	1.000	0.907	1.000	0.918
394 – 407	1.000	0.964	1.000	0.988	1.000	0.992
493 – 517	1.000	0.947	1.000	0.959	1.000	0.965

Table 9. Proteomic database results of parallel GP (null value means parallel computing does not give better result)

Cluster	Sequent GP		Parallel GP (2 islands)		Parallel GP (4 islands)	
	Se	***Sp***	***Se***	***Sp***	***Se***	***Sp***
1-3	1.000	0.909	1.000	0.926	1.000	0.942
4-7	1.000	0.922	1.000	0.965	1.000	0.983
10-14	1.000	0.944	1.000	0.982	-	-
17-19	1.000	0.718	1.000	0.882	1.000	0.956
20-23	1.000	0.965	1.000	1.000	-	-
1-7	1.000	0.901	1.000	0.989	-	-
17-23	0.900	0.714	1.000	0.905	1.000	0.952

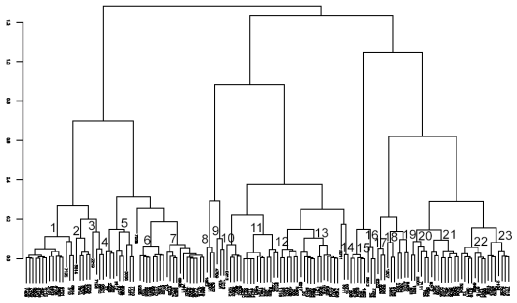

Figure 4. The dendrogram of proteomic database.

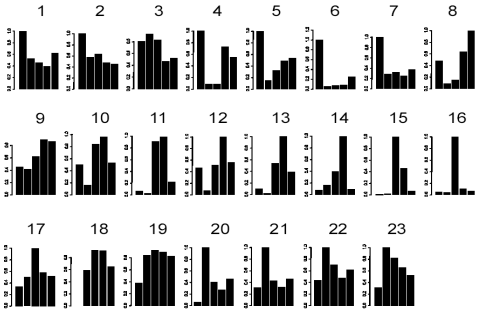

Figure 5. Average patterns of clusters of proteomic database.

Figure 6a. Cluster 263–296.

Figure 6b. Cluster 301–343.

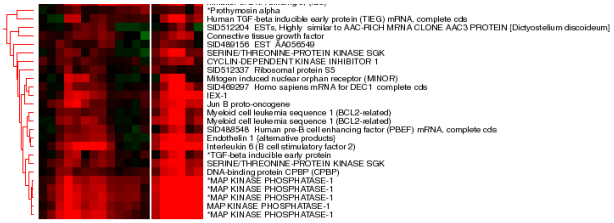

Figure 6c. Cluster 493–517.

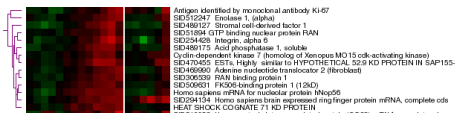

Figure 6d. Cluster 394–407.

Figure 6. Clusters of transcriptomic database.

5. ACKNOWLEDGMENT

We thank the anonymous referees for their pertinent suggestions. We also thank R. Nicolle for careful reading of the manuscript. This work is supported by the INCa (French National Institute of Cancer) through the INCa project PL-2010-196.

6. REFERENCES

[1] Khan, S.S. and Madden, M.G. 2009. A survey of recent trends in one class classification. *In Proceedings of the 20th Irish conference on Artificial intelligence and cognitive science*, 188–197.

[2] Tax, D. 2001. One class classification. *PhD thesis, Delft University of Technology*.

[3] Tax, D. and Duin, R. 1999. Data domain description using support vectors. *In Proceedings of ESAN99*, 251–256.

[4] Tax, D. and Duin, R. 1999. Support vector domain description. *Pattern Recognition Letters*, 20, 1191–1199.

[5] Scholkopf, B. and Smola, J.A. 2002. Learning with kernels. *MIT Press*.

[6] To, C. and Vohradsky, J. 2007. A parallel genetic algorithm for single class pattern classification and its application for gene expression profiling in streptomyces coelicolor. *BMC Genomics*, **8:**49.

[7] Yu, H. 2005. Single-class classification with mapping convergence. *Machine Learning*, 61(1), 49–69.

[8] De Comite, F., Denis, F., Gillerson, R., Letouzey, F. 1999. Positive and unlabeled examples help learning. *In Proceedings the 10th International Conference on Algorithmic Learning Theory*, 219-230.

[9] de Ridder, D., Tax, D., Duin, R. 1998. An experimental comparison of one-class classification methods. *In Proceedings of the 4th Annual Conference of the Advacned School for Computing and Imaging*.

[10] Manevitz, L. and Yousef, M. 2000. Document classification on neural networks using only positive examples. *In Proceedings of the 23rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, 304–306.

[11] Letouzey, F.D.F., and Gilleron, R. 2000. Learning from positive and unlabeled examples. *In Proceedings of Algorithmic Learning Theory, 11th International Conference*.

[12] Koza, J.R. 1992. Genetic Programming: On the programing of computers by means of natural selection. *MIT Press*, MA.

[13] Freitas, A.A. 2002. Data mining and knowledge discovery with evolutionary algorithms. *Springer Verlag*, Berlin.

[14] Cantu-Paz, E. 2001. Efficient and accurate parallel genetic algorithms. *Kluwer Academic Publishers*.

[15] Alba, E., Laguna, M., Luque, G. 2005. Workforce Planning with a Parallel Genetic Algorithm. *In Proceedings of the CEDI-MAEB'05*, 911–919.

[16] Calegari, P., Guidec, F., Kuonen, P., Kobler, D. 1997. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing: Special Issue on Parallel Evolutionary Computing*, Academic Press, 47(1), 86–90.

[17] Fernandez de Vega, F. 2005. Parallel genetic programming. *Workshop of the 2005 IEEE Congress on Evolutionary Computation*.

[18] Mangasarian, O.L., Wolberg, W.H. 1990. Cancer diagnosis via linear programming. *SIAM News* 23(5), 1-18.

[19] Street, W.N., Wolberg, W.H., Mangasarian, O.L. 1993. Nuclear feature extraction for breast tumor diagnosis. *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, 861–870.

[20] Peng, T., He, F., Zuo, W. 2006. Text classification from positive and unlabeled documents based on GA. *In Proceedings of the VECPAR*.

[21] Denis, F., Gilleron, R., Tommasi, M. 2002. Text classification from positive and unlabeled examples. *In Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*.

[22] Lee, W.S., and Liu, B. 2003. Learning with positive and unlabeled examples using weighted logistic regression. *In Proceedings of the 20th International Conference on Machine Learning*.

[23] Li, X., Liu, B., See-Kiong, Ng. 2010. Negative training data can be harmful to text classification. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[24] Kemmler, M., Rodner, E., Denzler, J. 2010. One-class classification with Gaussian processes. *In Proceedings of the 10th Asian conference on Computer vision*, 489–500.

[25] Hastie, T., Tibshirani, R., Firedman, J. 2003. The elements of statistical learning – data mining, inference, and prediction. *Springer*.

[26] Curry, R., and Heywood, M. 2007. One-class learning with multi-objective genetic programming. *In Proceedings of the 2007 IEEE Systems, Man and Cybernetics conference*, 1938–1945.

[27] Curry, R., and Heywood, M. 2009. One-class genetic programming. *In Proceedings of the European Conference on Genetic Programming*, 1–12.

[28] Grunenfelder, B., Rummel, G., Vohradsky, .J, Röder, D., Langen, H., Jenal, U. 2001. Proteomic analysis of the bacterial cell cycle. *Proc. Natl. Acad. Sci. USA*, 98, 4681–4686.

[29] Vohradsky,J., Janda, I., Grünenfelder, B., Berndt, P., Röder, D., Langen, H., Weiser, J., Jenal, U. 2003. Proteome of Caulobacter crescentus cell cycle publicly accessible on SWICZ server. *Proteomics*, 3, 1874-82.

[30] Iyer,V.R., Eisen, M.B., Ross, D.T., Schuler, G., Moore, T., Lee, J.C., Trent, J.M., Staudt, L.M., Hudson, J.Jr., Boguski, M.S., Lashkari, D., Shalon, D., Botstein, D., Brown, P.O. 1999. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283, 83–7.