2011

Gene Subset Selection Approaches Based on Linear Separability

Amirali Jafarian
University of Windsor

Gene Subset Selection Approaches Based on Linear Separability

by

Amirali Jafarian

A Thesis
Submitted to the Faculty of Graduate Studies
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2011

Your file Votre référence
ISBN: 978-0-494-81742-1
Our file Notre référence
ISBN: 978-0-494-81742-1

DECLARATION OF CO-AUTHORSHIO AND PREVIOUS PUBLICATION

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

This thesis also incorporates the outcome of a joint research undertaken in collaboration under the supervision of Professor Dr. Alioune Ngom. The collaboration is covered in Chapter 3 and 4 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the provision of advice when needed.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-authors to include the above materials in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes 3 original papers that have been previously published/submitted for publication in peer reviewed conferences, as follows:

Thesis Chapter	Publication title/full citation	Publication status
3 and 4	A New Gene Subset Selection Approach Based on Linear Separating Gene Pairs, IEEE International Conference on Computational Advances in Bio and medical Sciences (ICCABS 2011), Orlando FL, Feb 3-5, 2011, pp.105-110.	published
3 and 4	A Novel Recursive Feature Subset Selection Algorithm, 11th IEEE International Conference on Bioinformatics & Bioengineering, Taichung, Taiwan, October 24-26 2011	submitted
3 and 4	New Gene Subset Selection Approaches Based on Linearly Separating Genes and Gene-Pairs, the 6th International Conference on Pattern recognition in Bioinformatics, Delft, Netherlands, November 2-4 2011	submitted

I certify that I have obtained a written permission from the copyright owners to include the above published materials in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada

Copyright Act, I certify that I have obtained a written permission from the copyright owners to include such materials in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

We address the concept of linear separability of gene expression data sets with respect to two classes, which has been recently studied in the literature. The problem is to efficiently find all pairs of genes which induce a linear separation of the data. We study the Containment Angle (CA) defined on the unit circle for a linearly separating gene-pair (LS-pair) as an alternative to the paired *t-test* ranking function for gene selection. Using the CA we also show empirically that a given classifier's error is related to the degree of linear separability of a given data set. Finally we propose gene subset selection methods based on the CA ranking function for LS-pairs and a ranking function for linearly separation genes (LS-genes), and which select only among LS-genes and LS-pairs. Overall, our proposed methods give better results in terms of subset sizes and classification accuracy when compared to well-performing methods, on many gene expression data sets.

DEDICATION

To my beloved mother

for all your love, support and faith

I love you

ACKNOWLEDGEMENTS

First of all, I would like to convey my sincere gratitude to my supervisor, Dr. Alioune Ngom for his excellent supervision, advice and guidance throughout this thesis. Without his help and guidance this thesis would not have been completed. Also, he provided me unflinching encouragement and support in various ways.

Besides, I would also like to thank my internal reader, Dr. Luis Rueda, who has contributed in our research and gave us precious ideas to improve the quality of this research. Also, I would like to thank members of my master committee, Dr. Kemal Tepe, Department of Electrical and Computer Engineering, Dr. Luis Rueda, School of Computer Science, and Dr. Jianguo Lu, the chair of the committee for their constructive suggestions, helpful advices, and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

DNA microarrays give the expression levels for thousands of genes in parallel either for a single tissue, condition, or time point. In the former the snapshot of the expression of genes for different samples is taken, whereas the latter shows the expression for a period of time. Microarray data sets are also usually noisy with a low sample size given the large number of measured genes. Such data sets present many difficult challenges for sample classification algorithms. Since many genes are irrelevant to the target classes, considering them would not only introduce noise and degrade the classification performance, but also increase the computational time. Furthermore, with this huge number of genes, we have the problem of *curse of dimensionality* and the classification algorithms trained upon the data would be prone to the problem of *over-fitting*. The small number of samples makes it even worse. Hence in order to avoid the problems of *over-fitting* and *curse of dimensionality*, feature selection algorithms are applied to reduce the dimension of data to have at least faster and satisfactory classification accuracy with far less numbers of features selected. In addition, with filtering irrelevant genes, the biological information which was already hidden will be manifested. The feature subset selection problem is to find a smallest subset of genes, whose expression values allow sample classification with the highest possible accuracy; however Chen *et al.* [17] showed that finding the smallest feature subset selection is an NP-hard problem and some heuristic algorithms are needed to search for the optimal subset of genes. Feature subset selection methods have received considerable attention in recent years as better

dimensionality reduction methods than feature extraction methods, which yield features that are difficult to interpret. Many approaches have been proposed in the literature to solve this problem. A simple and common method is the *filter approach,* which first ranks single genes according to how well they each separate the classes and then selects the top r ranked genes as the gene subset to be used; where r is the smallest integer, which yields the best classification accuracy when using the subset. Also many feature ranking criteria are proposed based on different (or a combination of) principles, including *redundancy* and *relevancy* [2], [6]. Filter methods are simple and fast, but they do not necessarily produce the best gene subsets. Filter methods often deal with each gene separately and when each gene is considered individually, features' dependencies may be ignored, which may lead to unsatisfactory classification performance. Hence, in order to overcome this problem, a few *multivariate filter techniques* and *wrapper methods* have been introduced; *multivariate* filtering approaches can distinguish the target function better and model features' dependencies. Other methods introduced in literature are the *wrapper approaches*, which evaluate subsets of genes irrespective of any possible ranking over the genes. Such methods are based on heuristics which directly search the space of gene subsets and are guided by a classifier's performance on the selected gene subsets [9]. The best methods combine both gene ranking and wrapper approaches but they are computationally intensive.

Beside gene subset selection, in the context of microarray data many other analyses have been intensively studied, one of which is *clustering*; the problem in clustering is to find genes, which share similar patterns; the motivation of finding these genes is that they are functionality related.

1. Problem Statement

As already mentioned better approaches in filter methods are *multivariate* methods, modeling features' dependencies. However there are still only a few works for *multivariate* filtering approaches (eg., Selection of pairs of genes, triplets of genes or even group of genes) and most of the filtering approaches are categorized as *univariate* filtering approaches, where genes are ranked separately and the dependencies are ignored; therefore we believe gene subset selection, more specifically *multivariate* filtering approaches deserve more consideration.

In this thesis first we present gene subset selection methods based on the concept of *linear separability* of gene expression data sets as introduced recently in [1]. We use their geometric notion of *linear separation* by pairs of genes (where samples belong to one of two distinct classes termed *red* and *blue* samples in [1]) to define a simple criterion for selecting (best subsets of) genes for the purpose of sample classification. It has been suggested that an underlying molecular mechanism relates together the two genes of a separating pair to the phenotype under study, such as a specific cancer. Recently, some authors have considered pairs of genes as features to be used in filtering methods rather using than single genes. The motivation for using gene-pairs instead of single genes is that two single genes considered together may distinguish the classes much better than when they are considered individually; this is true even if one or both of the genes have low ranks from a ranking function defined for single genes. In other words, when we select only top-ranked single genes using such ranking function, some subsets of genes, which have greater class distinguishing capability (than the subset of top-ranked genes),

will not be selected due to the presence of low-ranked single genes. The authors of [2] devised the first gene selection method based on using pairs of genes as features. Given a gene-pair, they used *diagonal linear discriminant* (DLD) and compute the projected coordinate of each sample data on the DLD axis using only the two genes, and then take the two-sample *t*-statistic on these projected samples as the pair's score. The authors then devised two filter methods for gene subset selection based on the pair *t*-scores. Our approach in this thesis is to use both linearly separating single genes (LS-genes) and linearly separating gene-pairs (LS-pairs) as features for the purpose of finding the best gene subsets. We propose ranking criteria for both LS-genes and LS-pairs in order to evaluate how well such features separate the classes then devise methods that select among top-ranked LS-genes and LS-pairs.

Also as already mentioned *univariate* filter methods, which rank single genes according to how well they each separate the classes, are widely used for gene ranking in the field of microarray analysis of gene expression datasets. These methods rank all of the genes by considering all of the samples; however some of these samples may never be classified correctly by adding new genes and these methods keep adding redundant genes covering only some parts of the space and finally the returned subset of genes may never cover the space perfectly. In this thesis we also introduce a new gene subset selection approach which aims to add genes covering the space which has not been covered by already selected genes in a recursive fashion.

2. Outline

The rest of this document is organized as follows. Chapter 2 provides a literature review in the field of gene selection approaches. The fundamental part of this document is chapter 3, where our approaches for gene subset selection are described. Chapter 4 discusses the computational results obtained with our approaches and different experiments conducted to test the performance of our proposed approaches. Chapter 5 presents the details of datasets and pre-processing steps used in this thesis. Finally in chapter 6 we conclude and cite some possibilities for future works.

CHAPTER II

REVIEW OF LITERATURE

The main aim of this chapter is to provide the reader with a literature review of the previous important works done in the field of gene selection. Currently three major types of feature selection techniques, depending on how the feature selection search combines with the construction of the classification model, have been intensively employed in the field of gene selection and dimension reduction in microarray datasets. They are filter methods, wrappers methods, and embedded methods [18]. The three first sections of this chapter are categorized based on these three types of gene selection techniques mentioned, while the last part gives an overview of the recent work of Unger and Chor [1], who introduced the concept of *linear separability* of gene expression datasets.

1. Filter Methods

Filter methods attempt to select features based on intrinsic nature of the data. In these methods the gene selection process and classification process are separated; that is, first features' scores are calculated and then low-scoring features are filtered out, finally the remaining top ranked features will be used as input to machine learning classification algorithms. This kind of selection is faster, simpler and the selected genes give better generalization to unseen samples' classification [12]. Filter methods often treat mostly each gene separately and when each gene is considered individually, features' dependencies may be ignored, which may lead to unsatisfactory classification

performance [12]. Hence, in order to cope with this problem, a number of multivariate filter techniques and wrapper methods have been introduced; in the latter one (discussed in section 2 of this chapter) gene selection is directed by a classifier's performance in order to obtain the optimal subset of gene [9]. In this section, we discuss some univariate filter methods, followed by multivariate filter methods.

1.1. Univariate methods

As already mentioned univariate methods consider each gene separately and due to high dimension of microarray datasets, fast univariate techniques of filter methods have attracted the attention of many researchers.

1.1.1. *t-test*

One of the most used statistical filter methods is *t-test*. For a dataset S consisting of n features and m samples, the label of which is either +1 or -1 (2 class problem), the *t-test* criterion is calculated by the eq.1, in which for each gene the mean μ_i^+ (resp., μ_i^-) and the standard deviation σ_i^+ (resp., σ_i^-) of samples of positive class (resp., negative class) are used [22].

$$T(x_i) = \frac{|\mu_i^+ - \mu_i^-|}{\sqrt{\frac{(\sigma_i^+)^2}{n+} + \frac{(\sigma_i^-)^2}{n-}}} \qquad \text{(Eq.1)}$$

Also, in the eq.1, $n+$ (resp., $n-$) is the number of samples labeled as +1 (resp., -1).

1.1.2. Fisher Criterion

Similarly, Fisher's criterion [26] evaluates the degree of separation between two classes, being defined as follows:

$$J(g_i) = \frac{(\mu_i^+ - \mu_i^-)^2}{\sigma_i^{+2} + \sigma_i^{-2}} \quad \text{(Eq.2)}$$

Hedenfalk *et al.* [26] used *t*-test and Fisher Criteria in gene expression profiles of breast cancer. The experimental results showed 51 genes as the best that differentiated the three classes of tumours, returned by the Fisher criterion.

1.1.3. Signal to noise statistics

Golub *et al.* [4] introduced the modified ranking criterion called *Signal-to-noise statistics* (also called "*MIT correlation*"). Their modified ranking criterion is as follows:

$$MIT\,(x_i) = \frac{|\mu_i^+ - \mu_i^-|}{\sigma_i^+ + \sigma_i^-} \quad \text{(Eq.3)}$$

1.1.4. x^2 Statistics

Another example of statistical method used in microarray gene expression analysis is the work of Liu *et al.* [28]. In their work each gene is evaluated by measuring the *chi-squared* statistic with respect to classes:

$$x^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \quad \text{(Eq.4)}$$

Where m is the number of intervals; number of classes is shown by k; A_{ij} is the number of observations in the i_{th} interval, j_{th} class; the expected frequency E_{ij} is calculated by:

$$E_{ij} \frac{R_{i*} C_j}{N} \quad \text{(Eq.5)}$$

Where R_i is the number of observations in the i_{th} interval, C_j the number of observations in the j_{th} class and N the total number of observations.

Liu *et al.* [22] used this method for ranking genes. Experimental results with several statistics, included the *t-test*, indicate that this heuristics yields sometimes the most discriminatory features.

1.1.5. Relief Algorithm

Another univariate method to select relevant features is Relief algorithm, which does not depend on heuristics. The algorithm is very simple; that is, for each sample, the closest sample of a different class, called (*nearest miss*) and the closest sample of the same class, (*nearest hit*) are selected. Then the score of each feature is calculated as the average over all samples of magnitude of the difference between the distance to the *nearest hit* and the distance to the *nearest miss*, in the projection on the respective feature. Finally, Relief selects those features, whose scores, "*relevance level*", are above the given threshold τ [23].

Relief algorithm has been explored in gene selection by Wang and Makedon [27]; experimental results do not show outstanding results, although the performance of the algorithm is comparable with other algorithms.

1.1.6. T-NOM

TNOM [15] is an example of a simple univariate method introduced by Ben-Dor *et al.*(2000); they believe that informative genes have quite different values in the two classes (normal Vs. tumour), and by defining a threshold these two classes should be separated. Thus they set a threshold minimizing the number of training sample misclassification and define the number of errors, made based on the threshold, as quality of that gene and call it *"p-value"*. Finally genes are sorted according to their *p-values* and the highest ranked (with less number of errors) genes are selected. The idea of TNoM, also, was inspired by Sinha [24], who chose a classifier to find discriminative genes.

1.2. Multivariate methods

As already discussed, multivariate methods, which model feature dependencies, may distinguish the classes much better than univariate filter methods, where genes are considered individually; in other words, when we select only top-ranked single genes using a ranking function, some subsets of genes which have greater class distinguishing capability (than the subset of top-ranked genes) will be lost due to the presence of low-ranked single genes; it should be noted that multivariate methods are slower than

univariate methods but still much faster than wrapper methods. In this part we review some multivariate filter methods.

1.2.1. Minimum Redundancy and Maximum Relevancy (mRMR)

Most of the approaches in feature selection rank top genes according to their power of distinguishing between classes and select the top-ranked genes one by one at a time; however these techniques can bring along certain redundancy; Ding and Peng [6] believe while genes have high correlation to the target, they can be mutually far away from each other; hence, they proposed a very efficient feature selection method based on the minimum redundancy and maximum relevancy optimization approach; "Genes selected via mRMR provide a more balanced coverage of the space and capture broader characteristics of phenotypes" [6].

Let S be a subset of features or genes we are looking for. The minimum redundancy condition is given by:

$$min_s W_s, W_s = \frac{1}{|S|^2} \sum_{g_i, g_j \in S} I(g_i; g_j) \qquad \text{(Eq.6)}$$

Where g_i and g_j denote two genes. This expression tries to select all genes that are not correlated with each other, removing unnecessary genes, whose information could be expressed by other genes. Furthermore, to measure discriminative power of genes with respect to the target class, the following expression is used:

$$max_s W_s^C,\ W_s^C = \frac{1}{|S|} \sum_{g_i \in S} I(g_i; C) \qquad \text{(Eq.7)}$$

Where $I(g_i; C)$ is the MI between g_i and label or target class. Genes (features) are incrementally selected in order to optimize both (Eq.6) and (Eq.7) simultaneously.

They did extensive experiments on six different gene expression datasets and high accuracies were achieved based on their method, but all of their experiments were based on whole dataset not train and test procedure; that is, they select their genes on whole dataset and applied Leave-One-Out-Cross-Validation on whole of the dataset with only selected genes; however this set of experiment may inflate the results.

1.2.2. Pair based method based with *t*-test [2]

BØ and Jonassen in [2] proved that genes in pairs can present some useful information which is not discovered when genes are considered individually. They proposed two different methods for feature selection; in their fast method, first they select the top-ranked gene g_i and then find gene g_j such that the pair g_{ij} has maximal pair *t*-score on the DLD axis. In addition, they proposed another search strategy, which is more computationally expensive, but yielding better performance, which iteratively selects top disjointed ranked pairs. They experimented on two most used gene expression datasets Golub [4] and Alon [5] and got their highest accuracies by selecting only 15-30 genes.

2. Wrapper Methods

Different and better, but more computational expensive, methods are wrapper methods, which direct the gene selection by the performance of a classifier [9]. The most important

drawback of filter methods, which is the fact that they do not take into account the effect of the selected feature subset on the *posterior* performance of the classifier, has been solved by introducing wrapper methods. Hence in these methods a search algorithm is *"wrapped"* around the classification model. However it should be noted that wrapper approaches have been criticized for having a high risk of over-fitting [12]; in other words, the classifier performance is the only criterion in these methods and the gene selection is directed by the classifier's performance blindly on training data, which may give poor estimation and generalization on unseen data [12]. In addition, the computational time of wrapper methods is another reason why filter methods have been mostly favoured in microarray gene expression datasets; in other words, for each subset examined, a classifier is trained m times in a m-fold cross validation or B times in a B-bootstrap approach, which makes the wrapper methods very computational expensive.

Since the space of feature subsets grows exponentially with the number of genes increasing, heuristic search methods are exploited to guide the search for an optimal subset. Classical wrapper methods include forward selection and backward selection. Also, recently evolutionary based algorithms such as Genetic Algorithms (GA) have been introduced in microarray datasets as a better alternative than forward selection and backward selection algorithms.

2.1. Forward Selection and Backward Selections

In these strategies usually the greedy hill-climbing method is utilized to generate the subset of features. In Forward Sequential Selection, the search is started with an empty

subset of features (resp., in backward Sequential Search with all features selected) and add (resp., remove) one feature at a time (guided by the performance of a classifier) [20]; hence these algorithms avoid checking all possible subset of features to speed up the procedure but they do not find the optimal subset of genes. Instead of starting from an empty subset of features in Forward Selection or full subset of features in backward selection, the search can also be started with a randomly selected subset, and then forward search or backward search can be employed. Another approach is "plus-*l*-Minus-*r*" search strategy which adds (Resp,. removes) *n* features at a time instead of adding (Resp,. removing) one feature at a time [21].

2.2. Floating Search

As already mentioned classical wrapper methods, including *sequenctial backward selection* (SBS) and *sequenctial forward selection* (SFS) suffer from the so-called "nested effect"; consequently plus-*l*-Minus-*r* search strategy was introduced to overcome the problem of "nested effect" [21]; "plus-*l*-Minus-*r*" procedure consists of applying after each *l* forward steps *r* backward steps and then again *l* forward steps. The plus-*l*-Minus-*r* method needs the parameters "*l*" and "*r*" to be specified, whereas Sequential Floating Search identifies the number of forward steps (resp., backward steps) dynamically during the method's run by considering conditional inclusion and exclusion of features. That is, in the sequential Floating Forward Selection (SFFS), after each forward step a number of backward steps is considered, as long as the resulting subsets are better than previously evaluated one at that level [25].

2.3. TAFS Algorithm

Thermodynamic Annealing Feature Selection (TAFS) is a new algorithm for Feature Selection proposed by González *et al.* [29]. Given a suitable objective function, the algorithm uses *simulated annealing* technique to find a good subset of features maximizing the objective function. One of the advantageous of TAFS is its probabilistic capability to accept momentarily worse solution, which at the end may result in better hypotheses. In TAFS the notation of an $\mathcal{E}$-improvement was introduced; that is, a feature is accepted if it has a higher value of the objective function or a value not worse than $\mathcal{E}$%; this mechanism is taken into account for the noise in the evaluation of the objective function. Hence as well as the initial and final temperatures the value of $\mathcal{E}$ should be set.

González [29] also compared the performance of TAFS algorithm with the *Sequential Forward Floating Search* (SFFS) introduced by pudil *et al.* [25]; they showed that although TAFS selects more features, it achieves better performance. In addition, SFFS needs the desired subset size to be specified, which is difficult to estimate in many practical situations.

3. Embedded methods

The third category of feature selection is *embedded techniques,* in which the search for finding optimal subset of genes is embedded into the classifier construction. Hence, embedded techniques take advantage of: 1) including the interaction with a classifier to

find a subset with better performance, and 2) being less computational expensive than wrapper methods.

3.1. SVM-RFE

Guyon [14] introduced a new subset selection method, SVM Recursive Feature Elimination (*SVM-RFE*) for the purpose of gene selection, using the weights of the features in the SVM formulation to discard features with small weights. From the time of introducing SVM-RFE on, variant methods of this popular method have been proposed; one of these methods is the work of Mundra and Rajapakse [13], which is based on the concept of Support Vectors; they defined training samples as relevant (Support Vectors) and irrelevant data points (Non-Support Vectors) and they proved by considering only relevant data points they can get better results; although the *SVM–RFE* approach is very accurate and high classification accuracy is achieved in the experimental results of [14], it is quiet slow.

4. Linearly Separability of Gene Expression Datasets

Recently, [1] proposed a geometric notion of *linear separation* by gene pairs, in the context of gene expression data sets, in which samples belong to one of two distinct classes, termed *red* and *blue* classes. The authors then introduced a novel highly efficient algorithm for finding all gene-pairs that induce a linear separation of the two-class samples. Let $m = m_1 + m_2$ be the number of samples, out of which m_1 are red and m_2 are

blue. A gene-pair $g_{ij} = (g_i, g_j)$ is a *linearly separating* pair (LS-pair) if there exists a separating line L in the two-dimensional (2D) plane produced by the projection of the m samples according to the pair g_{ij}; that is, such that all the m_1 red samples are in one side of L and the remaining m_2 blue samples are in the other side of L, and no sample lies on L itself. Figures 1 and 2 show examples of LS and non-LS gene pairs, respectively.

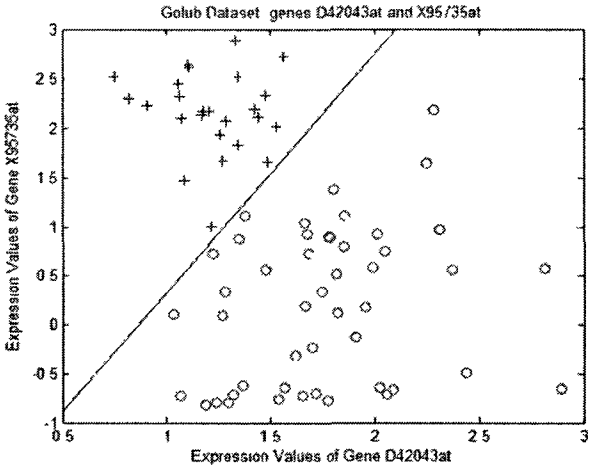


Figure 1. An LS pair taken from Golub (Leukemia) dataset.

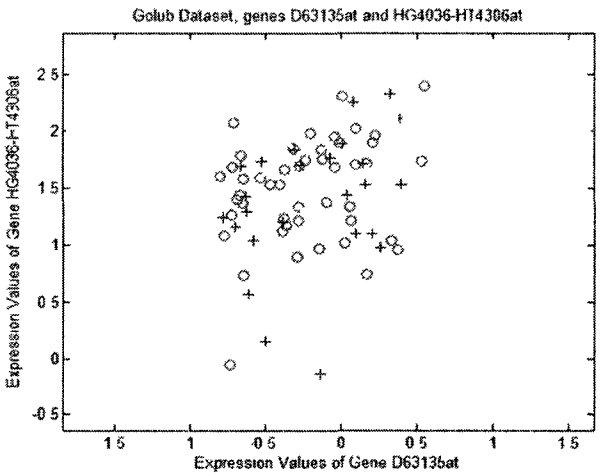


Figure 2. A non-LS pair taken from Golub (Leukemia) dataset.

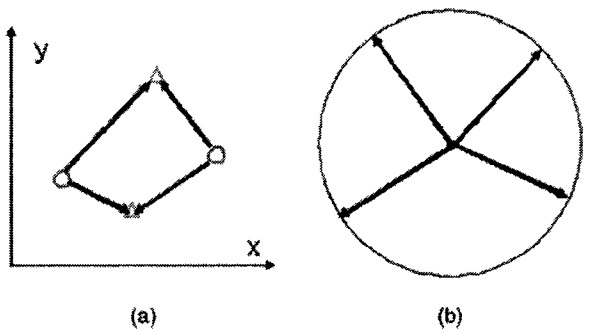


Figure 3. A set of four non-separable points. (a) The construction of the vectors. (b) Their projection onto the unit circle [1].

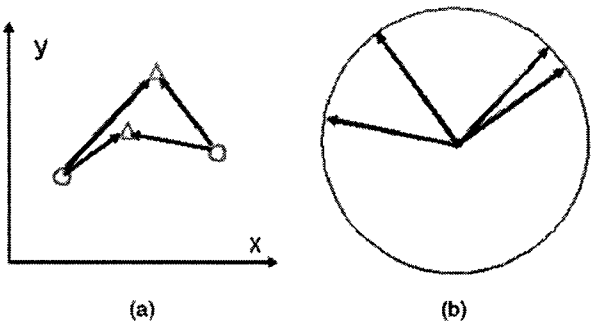


Figure 4. A set of four separable points producing vectors on the unit circle that are contained in a sector of angle $\beta < 180°$ [1].

In order to formulate a condition for linear separability, [1] first views the 2D points in a geometric manner. That is, each point of an arbitrarily chosen class, say red class, is connected by an arrow (directed vector) to every blue point. See Figures 3a and 4a, for example. Then the resulting m_1m_2 vectors are projected onto the unit circle, as in figures 3b and 4b, retaining their directions but not their lengths. The authors then proceed with a theorem proving that: *a gene pair* $g_{ij} = (g_i, g_j)$ *is an LS pair* if and only if *its associated unit circle has a sector of angle* $\beta < 180°$, *which contains all the* m_1m_2 *vectors*. Figures 3 and 4 illustrate this theorem for pairs (x, y). Thus, to test for linear separability of a pair

Table 1. Degree of separability of Datasets

Dataset Name	**Degree of Separability**
Small Beer	Very High
Squamous	Very High
Gordon	Very High
Bhattacharjee	Very High
Beer	High
Golub	High
Alon	Border Line
Adeno Beer	No

g_{ij} one only needs to find the vector with the smallest angle and the vector with the largest angle and check whether the two vectors form a sector of angle $\beta < 180°$ containing all $m_1 m_2$ vectors.

Using the theorem above, [1] proposed a very efficient algorithm for finding all LS-pairs of a data set. Next, they derived a theoretical upper bound on the *expected number* of LS- pairs in a *randomly labeled* data set. They also derived, for a given data set, an empirical upper bound resulting from shuffling the labels of the data at random. The degree to which an actual gene expression is linearly separable, (in term of the actual number of LS-pairs in the data) is then derived by comparing with the theoretical and empirical upper bounds. Seven out of the ten data sets, they have examined, were highly separable and very few were not (see Table 1).

4.1. Algorithm of Linear Separability [1]:

The complete algorithm to find LS-Pairs is as follows:

Initialize the min and max of right and left part of the unit circle as follows:

- Min right part of Unit Circle= INFINITY
- Min right part of Unit Circle= INFINITY
- Max right part of Unit Circle= -INFINITY
- Max right part of Unit Circle= -INFINITY

For each such pair of points $(p1_x, p1_y)$ and $(p2_x, p2_y)$:

1. Calculate deltaX $(p2_x - p1_x)$ and deltaY $(p2_y - p1_y)$.

2. Test whether they have the same X value (that is - whether deltaX==0), and if so - handle this case:

   2.a) If deltaY==0 - two points of both groups have exactly the same coordinates - no separation.

   2.b) If deltaY > 0 - set max refering to the right part of the unit circle, to INFINITY

   2.c) If deltaY < 0 - set min refering to the right part of the unit circle, to -INFINITY

3. Otherwise - calculate the slope of the line containing the vector. Please note that this is safe ONLY if deltaX != 0, thus *'step 2'* above was needed.

4. Handle 2 cases separately -

   1) deltaX < 0 - maintain min and max for a line representing the LEFT part of the unit circle.

   2) deltaX > 0 - maintain min and max for a line representing the RIGHT part of the unit circle.

5. Finally, and most importantly - test whether all the vectors so far can still be grouped into a <180 degrees wedge. If so - continue, otherwise - declare the pair of groups as non-separable. That is - if both maxima are at least as large as the minima of the other side of the unit circle (respectively) → no separation

CHAPTER III

PROPOSED METHODS

This chapter is intended to introduce our proposed methods in the field of gene subset selection. In the first part of this chapter we introduce our proposed methods for selecting subset of LS-genes and LS-pairs of genes and in the second part we propose our recursive feature subset selection algorithm, the aim of which is to cover the space which has not been covered by already selected genes.

1. Gene Subset Selection approaches based on Linearly Separating Genes and Pairs of Genes

In this method we use LS-genes and LS-pairs as features to select from, and for the purpose of finding a minimal number of such features such that their combined expression levels allow a given classifier to separate the two classes as much as possible. Our approach is to first obtain all the LS-genes and LS-pairs of a given data set, rank these features according to some ranking criteria, and then apply a filtering algorithm in order to determine the best subsets of genes.

1.1. LS-Pair Ranking Criterion

The LS pairs for given data sets were used as classifiers in [1], using a standard training-test process with cross-validation. The authors compared the performance of these new classifiers with that of an SVM classifier applied to the original data sets without gene

selection steps. They found that highly separable data sets exhibit low SVM classification errors, while low to non-separable data sets exhibit high SVM classification errors. However, no theoretical proof exists showing the relation between SVM performance and the degree of separability of a data set.

In this section, we study the relationship between the performance of a classifier applied to an LS pair of a given data set and the angle of the β-sector, discussed in chapter II (e.g, Fig. 4b). We call β, the *Containment Angle*. Intuitively, the smaller is β for an LS pair then the higher will be the accuracy of a classifier using the LS pair as input. That is, for LS pairs the generalization ability of the classifier decreases when β is close to 180°, since some samples from the two classes are very close to the separating line.

First, we used the algorithm of [1] to find all the LS pairs of a given data set. Second, we ranked the LS pairs in increasing order of their angles β; that is from small to large angles. For a data set D, we considered the top 10 LS pairs (i.e., smallest angles) and the bottom 10 LS pairs (i.e., largest angles), and then proceeded as follows. For each LS pair $g_{ij} = (g_i, g_j)$ of D, we applied a classifier with 10 runs of 10-fold cross-validation on D but using only g_i and g_j as features. We applied this to the separable data sets examined in [1]. The data were pre-processed in exactly the same manner as in [1]. Table 2 shows the results for 5 classifiers, *Diagonal Linear Discriminant* (DLD), *Support Vector Machine*, *k-Nearest Neighbour*, *Quadratic Diagonal Linear Discriminant* (QDA) and *SVM-Hard Margin.* An entry in columns B (resp., T) is the average of the classification accuracies on the bottom 10 (resp., top 10) LS pairs. Clearly, the accuracies in columns B are lower

than those in columns T. This enforces our intuition above while suggesting that one can use the Containment Angle as a measure of the quality of an LS pair.

Table 3 (resp., Table 4) shows the performance of SVM used on each of the top 3 (resp., bottom 3) LS pairs for each data set, and compares with SVM used on all genes of the data sets (last column). In Table 3, we can see that applying SVM on the best LS pairs yields at least better performance than on the full gene set, in majority of cases. Table 4 shows that LS pairs with largest containment angles β indeed yield worse classification performance than pairs, having smallest angles. Also, the accuracies increase (almost) monotonously in general from bottom to top LS pairs. There are few examples in Table 3, where there is a decrease of accuracy, say, from the second best pair to the best pair (see last row, for instance). These experiments also show that using LS pairs is a better alternative than using the full set of genes for sample classification purpose, since classifying using pairs is much faster than using the gene set while still giving satisfactory performances.

Table 2. Average of classifiers' perforances on bottom 10 (B) and top 10 (T) LS pairs.

	B DLD	T DLD	B SVM	T SVM	B KNN	T KNN	B QDA	T QDA	B SVM-Hard	T SVM-Hard
Beer	93 22%	98 96%	96 79%	98 96%	97 69%	98 96%	96 15%	98 96%	97 01%	98 96%
Small Beer	92 06%	97 94%	94 97%	98 99%	97 53%	98 96%	97 11%	98 49%	96 7%	98 96%
Golubl	94 42%	96 75%	94 85%	98 57%	93 93%	97 42%	93 32%	96 39%	95 72%	98 13%
Gordon	97 22%	98 37%	98 35%	98 86%	98 35%	99 45%	95 91%	97 81%	98 64%	99 43%
Squamous	92 59%	100%	93 07%	100%	93 73%	100%	94 44%	100%	92 59%	100%
Bhattacharjee	95%	95 2%	98 27%	99 08%	98 28%	99 33%	96 56%	97 19%	98 2%	99 4%

Table 3. Accuracy on the top three LS-pairs versus accuracy on the full gene set, using SVM with hard margin.

	TP1	TP2	TP3	Full Data
Small Beer	98 96%	98 96%	98 96%	**100%**
Beer	98 96%	98 96%	98 96%	**99.06%**
Squamous	**100%**	**100%**	**100%**	**100%**
Bhattacharjee	99 23%	**100%**	99 74%	98 08%
Gordon	99 83%	99 56%	**99.94%**	99 28%
Golub1	95 42%	**100%**	**100%**	98 61%

Table 4. Accuracy on the bottom 3 LS pairs versus accuracy on the full gene set, using SVM with hard margin.

	BP1	BP2	BP3	Full Data
Small Beer	96 88%	96 98%	96 15%	**100%**
Beer	96 46%	96 77%	97 08%	**99.06%**
Squamous	93 17%%	92 93%	92 68%	**100%**
Bhattacharjee	**98.21%**	98 01%	98 14%	98 08%
Gordon	98 78%	98 56%	98 45%	**99.28%**
Golub1	96 39%	96 11%	95 28%	**98.61%**

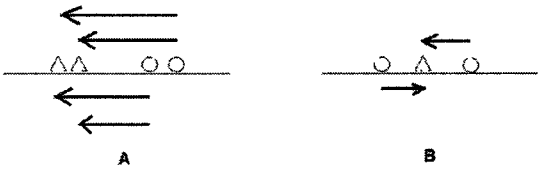


Figure 5. A set of points causing Linear Separability (Left Panel) Vs. Non Linear Separability (Right Panel)

Figure 6. The projection of vectors of LS Points in the Zero-Sphere (Left Panel) Vs. Non Linear Separability (Right Panel)

1.2. LS-Genes Ranking Criterion

In mathematics, an *n-sphere* is a generalization of the surface of an ordinary sphere to an arbitrary dimension. For a natural number, *n*, an *n-sphere* is defined as the set of points in (n+1)-dimensional Euclidean space. As an illustration, a *0-sphere* is a pair of points on a line, a *1-sphere* is a circle in the plane, and *2-sphere* is an ordinary sphere in three-dimensional space [30]. A single gene is an LS-gene if and only if all the m_1m_2 vectors in the corresponding *0-sphere* point are in the same direction (See Fig. 5 and 6 for a non LS-gene, a LS-gene and their projections in the *0-sphere*). We use a simple ranking criterion illustrated in Fig. 7: for each LS-gene, we compute the quantities A and B and use the ratio A/B as the score of the LS-gene.

1.3. Search strategies for selecting LS genes and LS pairs

Gene subset selection approaches based on gene pairs have been proposed in [2]. For a given gene pair, the authors used a two-sample *t*-statistic on projected data samples as the

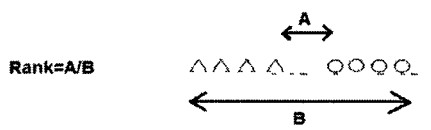


Figure 7. Ranking Criterion for LS Genes

score of pairs (pair t-score), and then pairs are ranked according to their t-scores for the purpose of subset selection. They devised two subset selection algorithms, which differ in the way gene pairs are selected for inclusion in a current subset. In their fastest method, they iteratively select the top-ranked gene g_i from the current list of genes, then find a gene g_j such that the t-score of the pair $g_{ij} = (g_i, g_j)$ is the maximum given all pairs $g_{ik} = (g_i, g_k)$, and then remove any other gene-pairs, containing either g_i or g_j; this continues until r genes are selected. In their best, but very slow method, they generate and rank all the possible gene pairs, and then select the top r ranked gene-pairs. The gene-pairs in [2] are not necessarily LS-pairs.

In this section, we propose gene subset selection approaches based on selecting only LS-genes and LS-pairs. The problem with this is that, initially, a data set may have a low degree of linear separability, and hence, not enough LS-Pairs to select from. To overcome this problem, we first apply SVM with soft margin on the initial given data set before performing any gene selection method, and then sort the support vector (SV) samples in decreasing order of their lagrange coefficients, obtained by training SVM; lagrange coefficients are non-zero for support vector samples, which are farthest from the separating maximum margin hyperplane and are probably misclassified. When there are no more LS-features to select from during the process of gene selection, we then iteratively remove the current SV sample having the largest Lagrange coefficient, until the resulting data set contains LS-features; we devised three filtering methods to be discussed below.

1.3.1. LS Approach

Our first gene subset selection method [11] proceeds by iteratively selecting disjoint LS-pairs until a subset S of r genes is obtained. The LS-pairs are ranked according to the ranking criterion, which is already discussed. Given a gene expression data set D, our first method is as follows:

LS: LS-Pair Selection on D:

1. $S \leftarrow \{\}$
2. $r \leftarrow$ desired number of genes to select
3. $d \leftarrow 0$
4. If $d < r$ Then
   a. $P \leftarrow$ set of LS-pairs of D
   b. $P \leftarrow P - \{g_{ij}$ s.t. $g_i \in S$ or $g_j \in S\}$
   c. Repeat
      i. $S \leftarrow S + \{g_{ij} \leftarrow$ top-ranked LS-pair in $P\}$
      ii. Apply a classifier on S and update *Best-S*
      iii. $P \leftarrow P - \{g_{ij}$ s.t. $g_i \in S$ or $g_j \in S\}$
      iv. $d \leftarrow d + 2$

      Until $d \geq r$ or $P = \{\}$
5. If $d < r$ Then
   a. Repeat
      i. $D \leftarrow D - \{$SV sample with largest Lagrange coefficient$\}$

      Until D contains LS-features
   b. Repeat from 4 with the resulting D
6. Return S, *Best-S*, and their performances

In the LS algorithm, S is the subset to be found, r is the desired size of S, and P is the sets of LS-pairs. In line 4.c.ii, we apply classifiers to the currently selected subset S to keep track of the best subset *Best-S* of size $\leq r$. We use ten runs of ten-fold cross-validation on S, and the algorithm returns subsets S and *Best-S* and their performances. SV samples with largest Lagrange coefficients are iteratively removed from data set D, in line 5.a.i, whenever there are not enough LS-pairs in the current D.

1.3.2. LSGP Approach

In the LS approach we only considered LS-pair, whereas our second gene subset selection method proceeds by iteratively, selecting in this order, from the set of LS-genes and then from the set of LS-pairs until a subset S of r genes is obtained. The LS-genes are ranked according to the ranking criteria discussed above. Given a gene expression data set D, our LSGP method is as follows:

LSGP: LS-Gene and LS-Pair Selection on D:

1. $S \leftarrow \{\}$
2. $r \leftarrow$ desired number of genes to select
3. $d \leftarrow 0$
4. $G \leftarrow$ set of LS-genes of D
5. $G \leftarrow G - \{g_i \text{ s.t. } g_i \in S\}$; '–' = *set-difference*
6. Repeat
   a. $S \leftarrow S + \{g_i \leftarrow$ top-ranked LS-gene in $G\}$
      ; '+' = *union*
   b. Apply a classifier on S and update *Best-S*
   c. $G \leftarrow G - \{g_i\}$
   d. $d \leftarrow d + 1$

   Until $d = r$ or $G = \{\}$
7. If $d < r$ Then
   a. $P \leftarrow$ set of LS-pairs of D
   b. $P \leftarrow P - \{g_{ij} \text{ s.t. } g_i \in S \text{ or } g_j \in S\}$
   c. Repeat
      i. $S \leftarrow S + \{g_{ij} \leftarrow$ top-ranked LS-pair in $P\}$
      ii. Apply a classifier on S and update *Best-S*
      iii. $P \leftarrow P - \{g_{ij} \text{ s.t. } g_i \in S \text{ or } g_j \in S\}$
      iv. $d \leftarrow d + 2$

      Until $d \geq r$ or $P = \{\}$
8. If $d < r$ Then
   a. Repeat
      i. $D \leftarrow D -$ {SV sample with largest Lagrange coefficient}

      Until D contains LS-features
   b. Repeat from 4 with the resulting D
9. Return S, *Best-S*, and their performances

The difference between LS algorithm and LSGP algorithm is in lines 4 up to 7, in which LSGP selects LS-genes. Moreover when a LS-gene g_t (resp., LS-pair g_{ab}) is selected, we also remove all LS-pairs containing g_t, (resp., g_a or g_b); see lines 7.b and 7.c.iii. This deletion is in order to minimize the redundancy. That is, when LS-gene g_t is selected then any LS-pair containing g_t will be redundant.

1.3.3. Graph-Based Methods

In [2] the authors first select the top-ranked gene g_i and then find a gene g_j such that the pair g_{ij} has maximal pair t-score. Also in their slow approach (which yields better performance than their fast method) they iteratively select the top-ranked pairs in such a way that the selected pairs are mutually disjoint from each other. That is, they delete all of those pairs which intersect the currently selected subset of genes. Assume a LS-pair $g_{ab} = (g_a, g_b)$ is selected and assume LS-pair $g_{bc} = (g_b, g_c) \in P$ not yet selected. If we remove g_{bc}, then the *possible* LS-triplet $g_{abc} = (g_a, g_b, g_c)$, which may yield a better subset S or a shorter subset *Best-S*, will be lost. Hence, we devised a third selection method for selecting LS-features.

Let G be the set of genes, we generalize the definition of linear separation to apply to any t-tuple $g_{i_1 \ldots i_t} = (g_{i1}, g_{i2}, \ldots, g_{it})$ of genes where $1 \leq t \leq |G|$, $1 \leq j \leq t$, and $i_j \in \{1, \ldots, |G|\}$, and say that: $g_{i_1 \ldots i_t}$ is a linearly separating t-tuple (LS-tuple) if there exists a separating (t-1)-dimensional hyperplane H in the t-dimensional sub-space defined by the genes in $g_{i_1 \ldots i_t}$. It remains open to generalize the theorem of [1] to t-tuples of genes, $t \geq 1$, by considering projecting the $m_1 m_2$ vectors obtained from the t-dimensional points onto a unit (t-1)-

sphere, and then determine a test for linearly separability of a t–tuple from the (t-1)-sphere. Clearly, the theorem is true for t=1: since a 0-sphere is a pair of points delimiting a line segment of diameter 2, and that the m_1m_2 vectors point in the same direction (i.e., they form a sector of angle 0) if and only the single gene is linearly separable. Therefore in our third method we consider the intersection graph $N = (P, E)$ where, the vertex set is the set of LS-pairs, P, in D and edges $(v_i, v_j) \in E$ if v_i and v_j have a gene in common. We then perform a graph traversal algorithm on N, which selects LS-pairs as the graph is being traversed. Given a gene expression data set D, our graph methods are as follows:

DF-LSGP: Graph-Based LSGP Selection on D

1. $S \leftarrow \{\}$
2. $r \leftarrow$ desired number of genes to select
3. $d \leftarrow 0$
4. $G \leftarrow$ set of LS-genes of D
5. $G \leftarrow G - \{g_i$ s.t. $g_i \in S\}$; '$-$' = *set-difference*
   ; *remove already selected LS-genes*
6. Repeat
   a. $S \leftarrow S + \{g_i \leftarrow$ top-ranked LS-gene in $G\}$
      ; '$+$' = *union*
   b. Apply a classifier on S and update *Best-S*
   c. $G \leftarrow G - \{g_i\}$
   d. $d \leftarrow d + 1$

   Until $d = r$ or $G = \{\}$
7. If $d < r$ Then
   a. $P \leftarrow$ set of LS-pairs of D
   b. $P \leftarrow P - \{g_{ij}$ s.t. $g_i \in G$ or $g_j \in G\}$
      ; *remove LS-pairs containing LS-genes*
   c. $P \leftarrow P - \{g_{ij}$ s.t. $g_i \in S$ and $g_j \in S\}$
      ; *remove already selected LS-pairs*
   d. Construct intersection graph $N = (P, E)$
   e. For each vertex g_{ij}: set *visited* $[g_{ij}] \leftarrow$ false
   f. While there are un-visited vertices and $d < r$ Do:
      i. *Stack* $\leftarrow \{\}$
      ii. $g_{ij} \leftarrow$ top-ranked vertex in N
      iii. Push g_{ij} onto *Stack*
      iv. While *Stack* $\neq \{\}$ and $d < r$ Do:
         1. Pop g_{ij} from *Stack*
         2. If g_{ij} is un-visited Then
            a. *visited*$[g_{ij}] \leftarrow$ true
            b. $d \leftarrow |S + \{g_{ij}\}|$
            c. $S \leftarrow S + \{g_{ij}\}$
            d. If S has changed Then
               i. Apply a classifier on S and update *Best-S*
            e. $P \leftarrow P - \{g_{ab}$ s.t. $g_a \in S$ and $g_b \in S\}$
               ; *delete already selected vertices from N*
            f. Push all un-visited neighbors of g_{ij} onto *Stack* starting from the least-ranked ones.
8. If $d < r$ Then
   a. Repeat
      i. $D \leftarrow D - \{$SV sample with largest Lagrange coefficient$\}$

      Until the resulting D contains LS-features
   b. Repeat from 4 with the resulting D
9. Return S, Best-S, and their performances

BF-LSGP: Graph-Based LSGP Selection on D

1. $S \leftarrow \{\}$
2. $r \leftarrow$ desired number of genes to select
3. $d \leftarrow 0$
4. $G \leftarrow$ set of LS-genes of D
5. $G \leftarrow G - \{g_i \text{ s.t. } g_i \in S\}$; '−' = *set-difference*
   ; *remove already selected LS-genes*
6. Repeat
   e. $S \leftarrow S + \{g_i \leftarrow$ top-ranked LS-gene in $G\}$
      ; '+' = *union*
   f. Apply a classifier on S and update *Best-S*
   g. $G \leftarrow G - \{g_i\}$
   h. $d \leftarrow d + 1$

   Until $d = r$ or $G = \{\}$
7. If $d < r$ Then
   a. $P \leftarrow$ set of LS-pairs of D
   b. $P \leftarrow P - \{g_{ij} \text{ s.t. } g_i \in G \text{ or } g_j \in G\}$
      ; *remove LS-pairs containing LS-genes*
   c. $P \leftarrow P - \{g_{ij} \text{ s.t. } g_i \in S \text{ and } g_j \in S\}$
      ; *remove already selected LS-pairs*
   d. Construct intersection graph $N = (P, E)$
   e. For each vertex g_{ij}: set *visited* $[g_{ij}] \leftarrow$ false
   f. While there are un-visited vertices and $d < r$ Do:
      i. *queue* $\leftarrow \{\}$
      ii. $g_{ij} \leftarrow$ top-ranked vertex in N
      iii. En-queue g_{ij} onto *queue*
      iv. While *queue* $\neq \{\}$ and $d < r$ Do:
         1. De-queue g_{ij} from *queue*
         2. If g_{ij} is un-visited Then
            a. $d \leftarrow |S + \{g_{ij}\}|$
            b. $S \leftarrow S + \{g_{ij}\}$
            c. If S has changed Then
               i. Apply a classifier on S and update *Best-S*
            d. $P \leftarrow P - \{g_{ab} \text{ s.t. } g_a \in S \text{ and } g_b \in S\}$
               ; *delete already selected vertices from N*
            e. En-queue all un-visited neighbors of g_{ij} onto *queue* starting from the high-ranked ones and change *visited*$[g_{ij}] \leftarrow$ true
8. If $d < r$ Then
   c. Repeat
      ii. $D \leftarrow D -$ {SV sample with largest Lagrange coefficient}

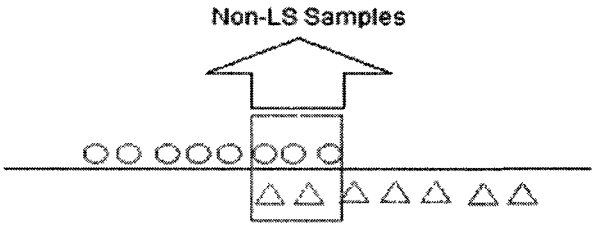
   Until the resulting D contains LS-features
   d. Repeat from 4 with the resulting D
9. Return S, Best-S, and their performances

The differences between our LSGP method and DF-LSGP are in lines 7. In DF-LSGP, the LS-genes are selected first as in the LSGP method. Then we iteratively select the best LS-pair vertex and its un-selected neighbors in a depth-first manner; see line 7.f and thereafter. This continues until the desired number of genes, r, is obtained. We have also implemented a breadth-first traversal of the graph, BF-LSGP, where the neighbors of a selected LS-pair are sent to a queue starting from the top-ranked ones. In practice, we do not create an intersection graph N (line 7.d) given that P may be very large for some data sets; we simply push or enqueue the top-ranked LS-pair from the initial P onto the stack or queue (line 7.f.iii) then simulate the graph-traversal algorithm.

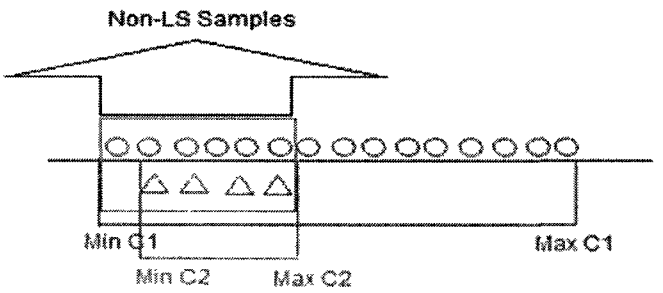
2. Recursive Feature Subset selection

Univariate filter methods, which rank single genes according to how well they each separate the classes, are widely used for gene ranking in the field of microarray analysis of gene expression datasets. These methods rank all of the genes by considering all of the samples; however some of these samples may never be classified correctly by adding new genes and these methods keep adding redundant genes covering only some parts of the sample space and the returned subset of genes may never cover the sample space perfectly. In this section we introduce a gene subset selection approach which aims to add genes covering the sample space which has not been covered by already selected genes in a recursive fashion.

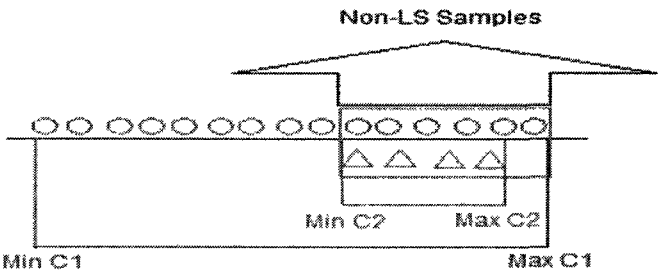
A. Samples of an LS Gene



B. Samples of a Non-LS Gene



C. Samples of a Non-LS Gene



D. Samples of a Non-LS Gene

Figure 8. Examples of LS-samples vs. Non-LS samples

Our algorithm, first selects gene g_i which can be selected by any gene ranking criteria and then partition samples to those causing non-linear separablility and those causing linear separablility based on the selected gene g_i; then the algorithm recursively selects gene g_j causing good degree of separation based only on non-LS samples. The motivation is that when some samples are linearly separable with gene g_i, they will still remain linearly separable by adding any other genes to gene g_i. Hence our algorithm focuses on those non-LS samples to find good degree of separation by adding gene g_j. Here first we introduce the definition of linear and non-linear separable samples and then we propose our recursive algorithm.

2.1. LS samples vs. non-LS samples

As said earlier we apply the ranking criterion on non-LS samples in a recursive fashion. We partition the samples to LS samples and non-LS samples as shown in figure 8; the intersection of classes is considered as non-LS samples (see fig 8.B for instance); however when the intersection of two classes' samples is one of the classes (see fig 8.C and 8.D for example), in this case, the non-LS samples are defined as follows:

If $(Max\,(C2) - Min\,(C1)) < (Max\,(C1) - Min\,(C2))$

non-LS Points= $Min\,(C1) \leq$samples$\leq Max\,(C2)$

Else

non-LS Points= $Min\,(C2) \leq$samples$\leq Max\,(C1)$

End

In the above equation, C1 (Res. C2) is the vector of samples of class1 (Resp. Class2). Hence in this case, e.g. fig 1.C and 1.D, we select those samples which are nearer to the border line as non-LS samples in order to have the greater cardinality of linear separability.

2.2. Recursive Feature Selection Algorithm

In this section, we propose our recursive gene subset selection approach based on partitioning samples to LS samples and non-LS samples; our approach is to select genes in such a way to cover the sample space better and broadly; our simple algorithm, however robust, consists of two loops; the inner loop recursively applies a ranking criterion on non-LS samples until the set of non-LS samples is empty or we have reached the desired number of genes. It, also, should be noted that when a gene is selected we apply machine learning classifiers on the subset of selected genes, so we keep track of the best subset of genes found so far. When there are not any more non-LS samples, the inner loop halts and all of the samples are considered for adding a new gene with the ranking criterion by re-starting the procedure. The complete algorithm is as follows:

Algorithm

1. $S \leftarrow \{\}$
2. $r \leftarrow$ desired number of genes to select
3. $d \leftarrow 0$
4. D=Data
5. Repeat
6. Repeat
    a. $G \leftarrow$Set of genes of D ranked according to a ranking function
    b. $G \leftarrow G - \{g_i$ s.t. $g_i \in S\}$; *'–' = set-difference*
        i. ; *remove already selected genes*
    c. $S \leftarrow S + \{g_i \leftarrow$ top-ranked LS-gene in $G\}$
        1. ; *'+' = union*
    d. Apply a classifier on S and update *Best-S*
    e. $d \leftarrow d + 1$
    f. *LS*= Linearly Separable Samples
    g. $D=D-$\{samples s.t s $\in$ LS\}
    h. Until $d = r$ or $D = \{\}$
7. *D=Data with whole samples*
8. Until *d=r*

In the algorithm, S is the subset to be found and r is the desired size of S, which has been set to 50 genes for all of our experimental results in this research. The algorithm starts with the full number of samples and if the gene selected does not cause linear separability for all of the samples then only non-LS samples will be considered for ranking and adding new genes (see line 6.f and 6.g where those LS-samples are deleted for gene ranking); in addition, when a gene is selected, we apply machine learning classifiers and keep track of the best subset (*Best-S*) achieved so far (See line 6.d). The inner loop iterates until the set of non-LS samples is empty or when it reaches the desired number of genes, r. If it has not reached the desired number of genes, r, and there are not any non-LS samples then the algorithm again starts ranking genes by considering all of

the samples (see line 7, where it adds all of the samples for gene ranking). Finally the algorithm returns subsets *S* and *Best-S* and their performances. The proposed algorithm is computationally efficient as well as being easy to implement.

2.2.1. Ranking criteria

As already mentioned our algorithm is flexible and can be implemented with any ranking criteria for selecting predictive genes based on non-LS data points; in this thesis we use two different ranking criteria; the first ranking criterion that we use is TNoM *"threshold number of misclassification"* [15] which ranks genes by the number of errors made by setting a threshold minimizing the number of training sample misclassification.

As the second ranking criterion we used Fisher's criterion [4], *f-test*, which evaluates the degree of separation between two classes' samples, For a dataset *S*, consisting of *n* features and *m* samples, the label of each is either +1 or -1 (2 class problem), the *f-test* criterion is calculated by the eq.8, in which for each gene the mean μ_i^+ (resp., μ_i^-) and the standard deviation δ_i^+ (resp., δ_i^-) of samples of positive class (resp., negative class) are used.

$$J(g_i) = \frac{(\mu_i^+ - \mu_i^-)^2}{{\sigma_i^+}^2 + {\sigma_i^-}^2} \qquad \text{(Eq.8)}$$

CHAPTER IV

COMPUTATIONAL EXPERIMENTS

This chapter aims to present computational experiments we have conducted based on the approaches introduced in chapter III. The Computational experiments in this chapter include extensive comparison with well-known approaches proposed so far in the literature. In the first part we compare our pair based approaches with *greedy-pair* method of [2]. In the second set of experiments we compared the performance of our pair based approaches with mRMR approach [6]. The third part of this chapter compares the performance of our four pair based selection approaches. The comparison of our recursive algorithm with its baselines and mRMR[6] is done in the fourth set of experiments. Finally, we present the result of our algorithms based on train and test procedure; that is, we select genes in training data then train the classifiers with training data but only with selected genes and test them on unseen data (test set).

1. Comparison of the pair based algorithms with the greedy pair method of [2]

In the first set of experiments, we compared our three filtering approaches (LSGP, DF-LSGP, and BF-LSGP) with the *greedy-pair* (GP) method of [2]. We compared on the two publicly available data sets (Golub [4] and Alon [5]) used in [2], which we have pre-processed in the same manner as in [2], and renamed as Alon2 and Golub2 to differentiate them with the Golub and Alon data sets used in [1] but pre-processed

differently. Alon2 has a very low degree of separability and Golub2 has a high degree of separability. We also compared with our *linearly separating pairs* (LS) method, but the results on Golub2 and Alon2 were exactly the same with LS and LSGP; thus the graph for LS is not shown in the ten figures below. In these experiments, we set the number of desired genes to $r = |S| = 50$ and also keep track of the best subset, *Best-S*, of size $\leq r$.

Figures 9 to 18 show the results of our three filtering methods compared with the *greedy-pair* method of [2]. SVM-Hard, SVM-Soft, KNN, DLD (*Diagonal Linear Discriminant*) and QDA (Quadratic Discriminate Analysis) classifiers were applied using ten runs of ten-fold cross-validation, and we returned the average accuracy over the hundred folds for both the subset S with size r and the best subset *Best-S*. The horizontal axis corresponds to the size of a selected gene subset and the vertical axis is the performance (classifier's accuracy) of the subset. [1] assigned each examined data set to a *degree of separability* class, that is *very high*, *high*, *borderline*, and *no* separability class, and Golub2 and Alon2 were respectively assigned to classes *high* and *borderline*. Naturally, the four filtering methods performed best on the *high separable* Golub2 data set (Fig. 9 to 13) and performed worst on the *borderline separable* Alon2 data set (Fig. 14 to 18). Our graph-based method, DF-LSGP and BF-LSGP performed better than LSGP and GP, in general; their curves are higher on average except with the DLD and QDA classifiers. LSGP performed the worst on average, except again with the DLD and QDA classifiers.

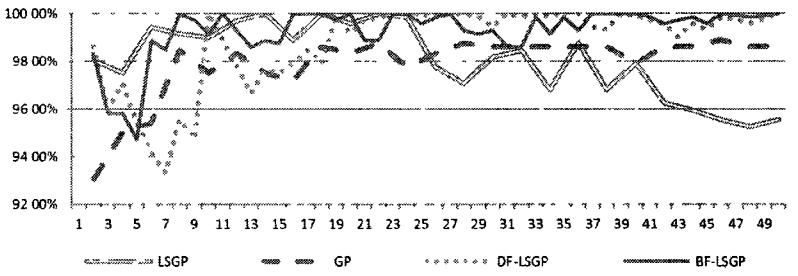


Fig 9. Performance of SVM-Hard on Golub2

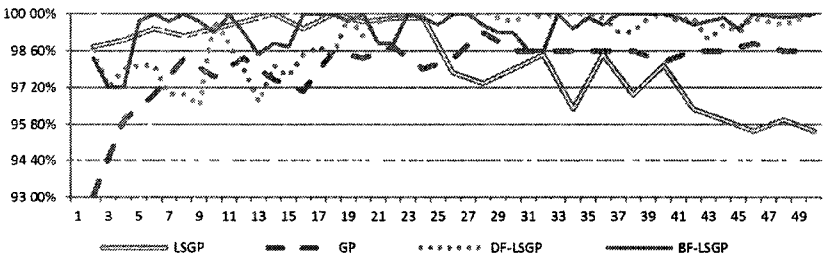


Figure 10. Performance of SVM-Soft on Golub2

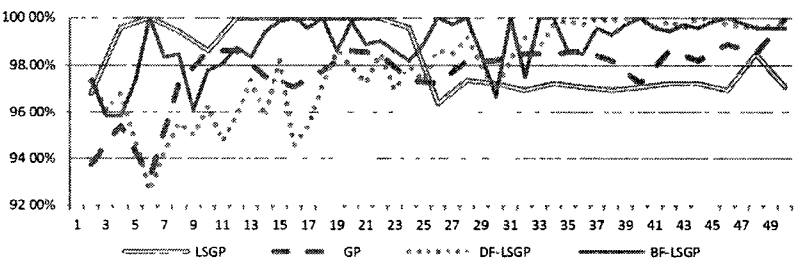


Figure 11. Performance of KNN on Golub2

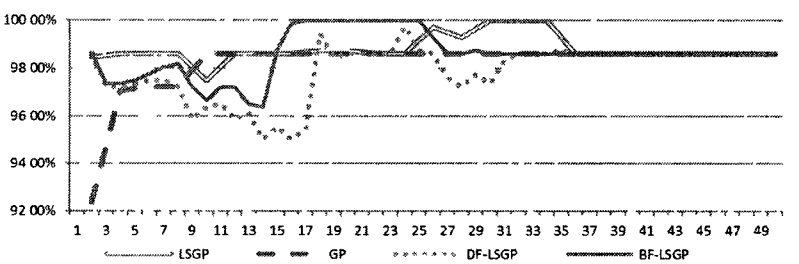


Figure 12. Performance of DLD on Golub2

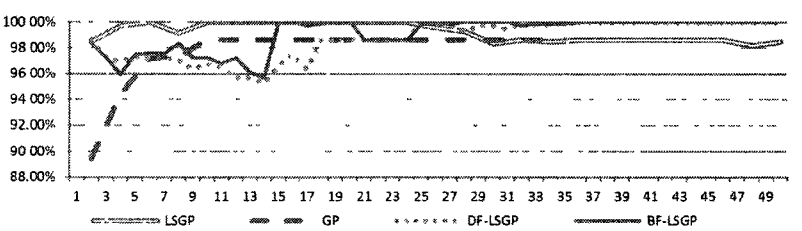


Figure 13. Performance of QDA on Golub2

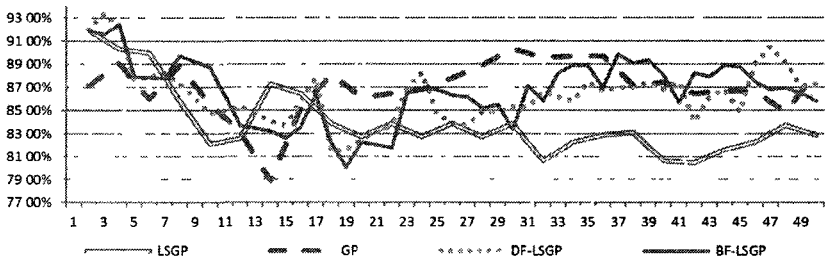


Figure 14. Performance of SVM-Hard on Alon2

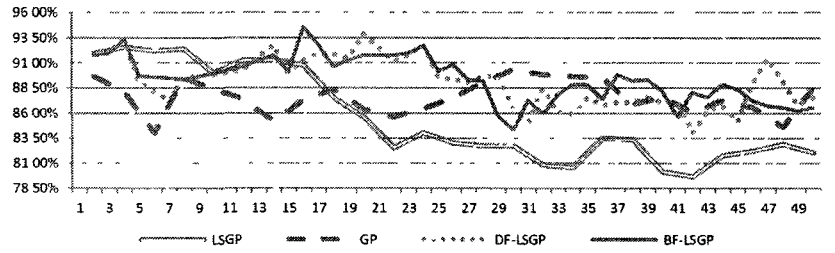


Figure 15. Performance of SVM-Soft on Alon2

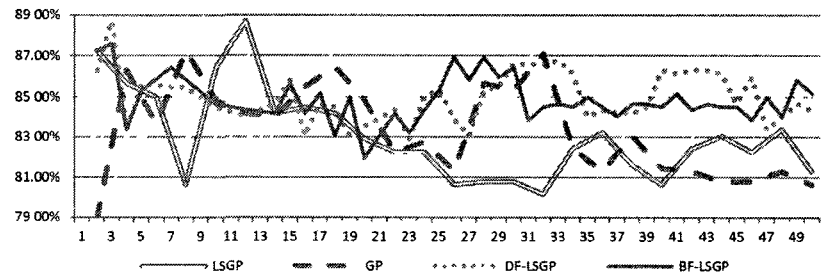


Figure 16. Performance of KNN on Alon2

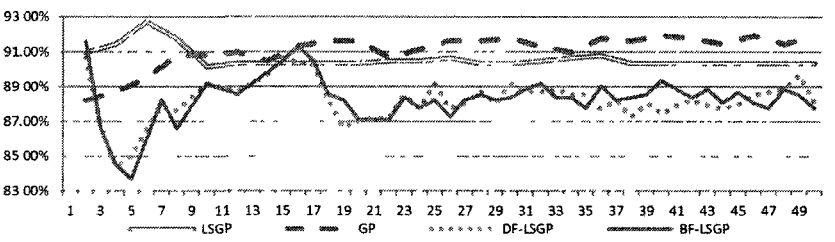


Figure 17. Performance of DLD on Alon2

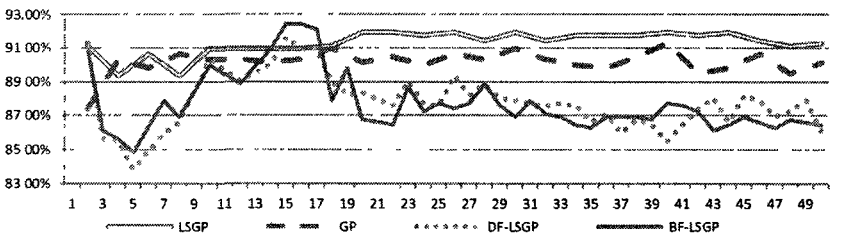


Figure 18. Performance of QDA on Alon2

The best subsets *Best-S* returned by our three methods are also smaller than those returned by GP. Our graph-based methods make use (and take advantage) of the information or knowledge already presented in the currently selected S subset in order to decide which LS-pairs to select next. Top-ranked LS-pairs which intersect S are always selected first, the advantage of which being the selection of t-tuples which are possibly linearly separating or which give better performances than arbitrarily selected LS-pairs. The selection of LS-pairs in GP and LSGP is somewhat arbitrary since it is based solely on their ranks.

Also we have compared the set of genes obtained with BF-LSGP approach, which gave us better performance in comparison to our other approaches; it is worth mentioning that genes reported by BF-LSGP have 32% and 28% similarity (percentage of genes in common) with the gene subsets reported by *greed-pair* method of [2] with Alon and Golub Datasets respectively.

2. Comparison of the pair based algorithm with mRMR [6]

In the second set of experiments, we compared our three LS-methods with the MIQ approach of mRMR [6], in which features are selected based on the *minimum redundancy and maximum relevancy* (mRMR) ranking criteria. The MIQ approach is among the best-performing selection methods in the literature, also it is computationally efficient as well as being easy to implement.

We experimented with eight data sets examined in [1]: *very high separable* data, Small Beer [7], Squamous [8], Gordon [10], Bhattacharjee [8]; *high separable* data, Beer [7], Golub [4]; *borderline separable* data, Alon [5]; and, *no separable* data, Adeno Beer [7]. Among these data sets, only Golub and Alon were used in [6]; thus we pre-processed them as in [6] for a fair comparison, by normalizing to zero-mean and unit-variance and renamed as Golub3 and Alon3. The remaining data sets are pre-processed as in [1]. Additionally for MIQ only, we discretized the data sets into three states as in [6]. Our methods do not require discretization of the data. As in [6], we applied Leave-one-out cross-validation for each data set with classifiers, and then we returned the average performances of the best subsets, *Best-S*, found along with their sizes.

Given the eight data sets and different classifiers, Table 5 and 6 show the average

Table 5. [XX]-LSGP versus MIQ [6] on subsets S

	SVM-Soft				DLD				KNN				SVM-Hard			
	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP
Beer	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Small Beer	100%	100%	98 96%	98 96%	100%	98 96%	97 92%	97 92%	100%	100%	98 96%	98 96%	100%	100%	100%	100%
Squamous	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Gordon	-	100%	100%	100%	-	98 90%	98 90%	98 90%	-	99 45%	100%	100%	-	100%	100%	99 45%
Bhattacharjee	99 36%	99 36%	99 36%	100%	99 36%	99 36%	99 36%	98 08%	98 72%	99 36%	99 36%	100%	99 36%	99 36%	100%	99 36%
Golub3	100%	98 61%	100%	100%	100%	98 61%	98 61%	98 61%	100%	97 22%	100%	100%	100%	98 61%	100%	100%
Alon3	80 65%	83 87%	93 55%	93 55%	87 10%	88 71%	90 32%	90 32%	83 87%	77 42%	79 03%	79 03%	80 65%	83 87%	93 55%	93 55%
Adeno Beer	94 19%	79 07%	77 91%	77 91%	94 19%	87 21%	86 05%	86 05%	95 35%	79 07%	88 37%	88 37%	94 19%	73 26%	82 56%	82 56%

accuracies of *S* and *Best-S* respectively, and the sizes (in parenthesis) of *Best-S* found by MIQ, LSGP, DF-LSGP, and BF-LSGP. As expected, all four selections methods performed worst on the difficult data sets, Alon3 and Adeno Beer, which are classified as *borderline* and *no separable* [1], respectively; the subsets *Best-S* are larger and have lower performance than those obtained from the *high* to *very high separable* data sets.

In general, our three LSGP methods compare very well with MIQ; sometimes they give better accuracies but with a bit larger subset or they give smaller subsets for not much different accuracies. Also as expected, the graph-based methods performed better than the simple LSGP method.

Table 6. [XX]-LSGP versus MIQ [6] on Best-S

	SVM-Soft				DLD				KNN				SVM-Hard			
	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP	MIQ	LSGP	DF-LSGP	BF-LSGP
Beer	100% (2)	100% (4)	100% (4)	100% (4)	100% (50)	100% (4)	100% (4)	100% (4)	100% (2)	100% (4)	100% (4)	100% (4)	100% (2)	100% (4)	100% (4)	100% (4)
Small Beer	100% (2)	100% (5)	100% (5)	100% (5)	100% (23)	98 96% (1)	98 96% (1)	98 96% (1)	100% (2)	100% (4)	100% (4)	100% (4)	100% (2)	100% (4)	100% (4)	100% (4)
Squamous	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)
Gordon	-	100% (4)	100% (3)	100% (3)	-	99 45% (10)	100% (28)	98 90% (35)	-	100% (4)	100% (7)	100% (10)	-	100% (2)	100% (2)	100% (2)
Bhattacharjee	99 36% (2)	100% (2)	100% (2)	100% (2)	99 36% (7)	99 36% (4)	99 36% (18)	99 36% (18)	99 36% (2)	99 36% (2)	100% (3)	100% (3)	99 36% (2)	99 36% (2)	100% (3)	100% (3)
Golub3	100% (6)	100% (4)	100% (47)	100% (20)	100% (20)	98 61% (32)	98 61% (4)	98 61% (4)	100% (2)	100% (4)	100% (50)	100% (50)	100% (6)	100% (4)	100% (47)	100% (20)
Alon3	88 71% (7)	96 77% (10)	96 77% (11)	96 77% (11)	88 71% (7)	88 71% (16)	90 32% (12)	90 32% (12)	90 32% (11)	88 71% (2)	88 71% (2)	88 71% (2)	85 48% (8)	93 55% (2)	95 16% (36)	93 55% (2)
Adeno Beer	95 35% (26)	89 53% (10)	90 70% (18)	90 70% (18)	95 35% (16)	89 53% (22)	89 53% (14)	89 53% (14)	95 35% (50)	90 70% (34)	90 70% (47)	90 70% (46)	96 51% (38)	84 88% (14)	88 37% (18)	88 37% (18)

3. Comparison between our pair based approaches

In our third set of experiments, we study the performances of LS [11] and LSGP methods only on data sets containing LS-genes. Only LS-pairs were considered in our linearly separating (LS) selection algorithm in LS approach [11]. We compared both LS with our LSGP methods on data sets containing LS-genes and LS-pairs, in order to study their performances given some classifiers. Among the data sets listed in Table 29, only the three separable data sets, Beer [7], Small Beer [7] and Squamous [8] contain LS-genes. Only Beer has a *high* degree of separability and the remaining two data sets are *very high*ly separable, thus we decided to experiment with two data sets of different categories which are Beer and Small Beer. We performed ten runs of ten-fold cross-validation and returned the performances of the best subsets found by LS and our LSGP methods. Table

Table 7. [XX]-LSGP versus LS [11] on Best-S's

	LS		LSGP		DF-LSGP		BF-LSGP	
	Beer	S. Beer	Beer	S. Beer	Beer	S. Beer	Beer	S. Beer
SVM-Soft	100% (6)	100% (10)	100% (4)	100% (5)	100% (4)	100% (5)	100% (4)	100% (5)
KNN	100% (6)	100% (10)	100% (4)	100% (4)	100% (4)	100% (4)	100% (4)	100% (4)
DLD	100% (6)	98 95% (2)	100% (4)	98.95% (1)	100% (4)	98.95% (1)	100% (4)	98.95% (1)
QDA	100% (12)	100% (14)	100% (8)	100% (9)	100% (8)	100% (8)	100% (8)	100% (8)

7 shows the results for this set of experiments with classifiers; as expected, our new methods give smaller subsets than LS approach, which has some degree of redundancy.

4. Comparison of the Recursive Algorithm with baselines and mRMR [6]

In the third set of experiments, we compared our recursive algorithm implemented with two different ranking criteria mentioned earlier with their baselines and the MIQ approach of mRMR [6]. We experimented with the same eight data sets preprocessed in the same manner of [1] except Golub and Alon, which were used in [6]; thus we pre-processed them as in [6]. We applied Leave-One-Out-Cross-Validation for each data set and we experimented with SVM-Soft, KNN, DLD (*Diagonal Linear Discriminant*) and SVM-Hard classifiers, and then we returned performances of the subsets, S of size 50 (See Tables 8 and 10). Also Tables 9 and 11 show the performances of the best subsets, *Best-S*, found along with their sizes.

By looking at Tables 8 to 11, the significant improvement of our recursive algorithm (which is shown by Rec-[XX], where XX is a ranking criterion) in comparison with their baselines is noticeable. For those datasets that *f-test* and TNoM give us low accuracies and there is enough space to improve, the dramatic improvement of our recursive algorithm is observed, whereas for those dataset that *f-test* and TNoM return subsets with the high accuracies (ie. near to 100% accuracy) and there is not enough space to improve anymore, the improvement of our algorithm is trivial. Also our results are completely comparable with mRMR approach.

Table 8. Comparison of Recursive Algorithm implemented by f-test with baselines based on subsets S

	SVM-Soft			DLD			KNN			SVM-Hard		
	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test
Beer	100%	98 96%	100%	100%	100%	100%	100%	98 96%	100%	100%	100%	100%
Small Beer	100%	98 96%	98 96%	100%	98 96%	98 96%	100%	98 96%	100%	100%	98 96%	100%
Squamous	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Gordon	-	98 90%	100%	-	99 45%	99 45%	-	99 45%	99 45%	-	99 45%	99 45%
Bhattacharjee	99 36%	98 08%	99 36%	99 36%	98 72%	99 36%	98 72%	97 44%	99 36%	99 36%	98 72%	98 72%
Golub3	100%	98 61%	98 61%	100%	97 22%	97 22%	100%	95 83%	98 61%	100%	98 61%	98 61%
Alon3	80 65%	80 65%	87 10%	87 10%	85 48%	90 32%	83 87%	75 81%	85 48%	80 65%	80 65%	85 48%
Adeno Beer	94 19%	84 88%	94 19%	94 19%	87 21%	91 86%	95 35%	84 88%	87 21%	94 19%	84 88%	96 51%

Table 9. Comparison of Recursive Algorithm implemented by f-test with baselines based on Best-S

	SVM-Soft			DLD			KNN			SVM-Hard		
	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test	MIQ	f-Test	Rec-f-Test
Beer	100% (2)	100% (2)	100% (2)	100% (50)	100% (2)	100% (2)	100% (2)	100% (2)	100% (2)	100% (2)	100% (2)	100% (2)
Small Beer	100% (2)	98 96% (1)	100% (3)	100% (23)	98 96% (1)	98 96% (1)	100% (2)	100% (6)	100% (3)	100% (2)	100% (3)	100% (3)
Squamous	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)	100% (1)
Gordon	-	100% (8)	100% (5)	-	99 45% (5)	100% (8)	-	100% (9)	100% (5)	-	100% (7)	100% (6)
Bhattacharjee	99 36% (2)	98 72% (1)	99 36% (8)	99 36% (7)	99 36% (9)	99 36% (21)	99 36% (2)	98 72% (23)	99 36% (14)	99 36% (2)	98 72% (49)	99 36% (6)
Golub3	100% (6)	98 61% (40)	100% (43)	100% (20)	97 22% (10)	97 22% (28)	100% (2)	97 22% (32)	100% (12)	100% (6)	98 61% (40)	100% (43)
Alon3	88 71% (7)	88 71% (9)	95 16% (19)	88 71% (7)	88 71% (15)	90 32% (33)	90 32% (11)	83 87% (15)	91 94% (15)	85 48% (8)	88 71% (5)	90 32% (23)
Adeno Beer	95 35% (26)	88 37% (10)	96 51% (19)	95 35% (16)	90 70% (7)	91 86% (15)	95 35% (50)	91 86% (10)	95 35% (30)	96 51% (38)	88 37% (9)	96 51% (49)

Table 10. Comparison of Recursive Algorithm implemented by TNoM with baselines based on subsets S

	SVM-Soft			DLD			KNN			SVM-Hard		
	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM
Beer	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	98 96%
Small Beer	100%	98 96%	100%	100%	98 96%	98 96%	100%	98 96%	98 96%	100%	100%	100%
Squamous	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Gordon	-	98 90%	99 45%	-	98 90%	99 45%	-	99 45%	99 45%	-	98 90%	99 45%
Bhattacharjee	99 36%	98 08%	98 72%	99 36%	98 08%	98 72%	98 72%	98 08%	98 72%	99 36%	98 08%	98 72%
Golub3	100%	95 83%	98 61%	100%	97 22%	97 22%	100%	94 44%	98 61%	100%	95 83%	98 61%
Alon3	80 65%	80 65%	87 10%	87 10%	87 10%	87 10%	83 87%	83 87%	80 65%	80 65%	77 42%	87 10%
Adeno Beer	94 19%	84 88%	93 02%	94 19%	84 88%	90 70%	95 35%	89 53%	84 88%	94 19%	84 88%	95 35%

Table 11. Comparison of Recursive Algorithm implemented by TNoM with baselines based on Best-S

	SVM-Soft			DLD			KNN			SVM-Hard		
	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM	MIQ	TNoM	Rec-TNoM
Beer	100% (2)	100% (4)	100% (4)	100% (50)	100% (6)	100% (6)	100% (2)	100% (3)	100% (3)	100% (2)	100% (3)	100% (3)
Small Beer	100% (2)	100% (4)	100% (4)	100% (23)	98.96% (2)	98 96% (2)	100% (2)	100% (3)	100% (3)	100% (2)	100% (3)	100% (3)
Squamous	100% (1)	100% (2)	100% (2)	100% (1)	100% (2)	100% (2)	100% (1)	100% (2)	100% (2)	100% (1)	100% (2)	100% (2)
Gordon	-	99 45% (6)	99 45% (5)	-	98 90% (10)	99 45% (50)	-	100% (41)	100% (6)	-	99 45% (6)	100% (5)
Bhattacharjee	99 36% (2)	98 72% (1)	99 36% (3)	99 36% (7)	99 36% (4)	99 36% (7)	99 36% (2)	99 36% (2)	99 36% (3)	99 36% (2)	99 36% (2)	99 36% (27)
Golub3	100% (6)	98 61% (32)	98 61% (15)	100% (20)	97 22% (34)	97 22% (4)	100% (2)	97 22% (4)	98 61% (20)	100% (6)	98 61% (32)	98 61% (15)
Alon3	88 71% (7)	87 10% (24)	96 77% (33)	88 71% (7)	87 10% (3)	88 71% (12)	90 32% (11)	85 48% (34)	85 48% (7)	85 48% (8)	88 71% (12)	96 77% (33)
Adeno Beer	95 35% (26)	96 51% (20)	96 51% (15)	95 35% (16)	88 37% (23)	94 19% (39)	95 35% (50)	94 19% (28)	91 86% (14)	96 51% (38)	96 51% (21)	95 35% (34)

5. Train and Test Procedure

It should be noted that in the previous experiments, we used the entire data set: 1) to rank and select genes; 2) to derive a subset *S* (or *Best-S*) of genes; and then 3) we used cross-validation to estimate the classification accuracy using only the selected subset. We performed another set of experiments, in which the ranking and subset selection are performed on the training dataset within the framework of ten-fold cross-validation process. That is, we partition a data set *D* into ten distinct parts, and in each iteration of ten-fold cross validation process: 1) we perform feature ranking on the nine-part training set; 2) train a classifier on this training set but using only the selected genes; and 3) estimate the performance of classification on the remaining one-part validation set. We did this set of experiments with our methods on the eight data sets of [1] (given in Table 29) and which are pre-processed as in [1] also. The results for these experiments are shown in Tables 12 to 15. We show the performances of our algorithms in terms of average accuracy for both subsets *S* and *Best-S* with SVM-Soft, KNN and SVM-Hard classifiers. For our recursive algorithms we also compared their performance with their baselines (See Tables 14 and 15). We must note that since feature ranking and selection is performed in each fold of the ten-fold cross-validation, then ten different subsets *S* and *Best-S* are obtained after the ten iterations of the cross-validation process.

So since in this set of experiments subsets are not fixed as in our previous sets of experiment above for subsets *Best-S*, in Tables 13 and 15, we list in parenthesis the minimum, the average, and the maximum size of the hundred subsets *Best-S* obtained after the ten runs of ten-fold cross-validation, beside showing the average of the accuracies of the hundred subsets. For subsets *S*, an entry is the average of the accuracies of the hundred subsets of size $r = 50$ each. The averages in Tables 12 to 15 are quite high, even for the least separable data sets Alon and Adeno Beer. Also, by looking at Tables 14 and 15 we see that our recursive algorithm is at least comparable with the baselines. In

Table 12. Accuracy of S for [XX]-LSGP, with ranking and selection on training sets.

	KNN			SVM-Soft			SVM-Hard		
	LSGP	DF-LSGP	BF-LSGP	LSGP	DF-LSGP	BF-LSGP	LSGP	DF-LSGP	BF-LSGP
Beer	99.26%	99.07%	98.94%	99.47%	99.18%	98.94%	99.69%	99.28%	100%
Small Beer	98.98%	98.96%	98.96%	98.98%	98.96%	98.96%	99.08%	99.27%	98.96%
Squamous	100%	100%	100%	100%	100%	100%	100%	100%	100%
Gordon	99.06%	99.09%	99.14%	99.23%	99.02%	98.89%	98.78%	98.72%	98.90%
Bhttacharjee	98.29%	97.33%	96.18%	98.29%	97.65%	96.96%	97.82%	97.63%	97.09%
Golub	95.89%	95.32%	95.31%	96.11%	93.92%	95.23%	96.84%	96.26%	95.99%
Alon	84.57%	85.95%	81.95%	80.57%	80.17%	79.95%	78.45%	80.55%	82.86%
Adeno Beer	75.04%	76.80%	74.83%	74.23%	75.47%	76.29%	73.84%	76.91%	76.17%

addition, for all data sets, we obtained a subset *Best-S* with the maximal accuracy of 100%.

Table 13. Accuracy of Best-S for [XX]-LSGP, with ranking and selection on training sets.

	KNN			SVM-Soft			SVM-Hard		
	LSGP	DF-LSGP	BF-LSGP	LSGP	DF-LSGP	BF-LSGP	LSGP	DF-LSGP	BF-LSGP
Beer	100% (1, 2 31, 36)	99 80% (1, 2 46, 34)	99 78% (1, 2 06, 21)	100% (1, 2 36, 13)	99 60% (1, 1 81, 11)	99 90% (1, 2 16, 18)	100% (1, 2 16, 18)	99 90% (1, 2 66, 34)	100% (1, 2 42, 21)
Small Beer	99 18% (1, 1 21, 3)	98 96% (1, 1 18, 3)	98 96% (1, 1 21, 3)	99 18% (1, 1 81, 3)	98 96% (1, 1 13, 2)	98 96% (1, 1 15, 3)	99 69% (1, 3 77, 32)	99 27% (1, 1 90, 47)	99 07% (1, 1 66, 46)
Squamous	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)	100% (1, 1, 1)
Gordon	99 61% (2, 3 76, 28)	99 70% (2, 4 58, 43)	99 77% (2, 4 60, 47)	99 56% (2, 4 12, 30)	99 32% (2, 4 15, 44)	99 52% (2, 4 55, 40)	99 50% (2, 3 88, 44)	99 32% (2, 3 95, 37)	99 84% (2, 4 60, 40)
Bhttacharjee	98 81% (1, 3 41, 14)	98 36% (1, 2 97, 31)	98 19% (1, 3 15, 44)	98 68% (1, 2 68, 16)	98 29% (1, 2 43, 32)	98 11% (1,2 48, 44)	98 61% (1, 3 06, 18)	98 29% (1, 2 79, 26)	98 18% (1,3 10, 46)
Golub	98 01% (2, 5 41, 42)	98 46% (2, 7 79, 43)	97 65% (2, 6 14, 45)	97 70% (2, 4 65, 48)	97 40% (2,4 96, 40)	97 61% (2, 5 83, 45)	98 67% (2, 5 35, 30)	99 11% (2, 6 1, 49)	98 86% (2, 6 71, 48)
Alon	93 43% (2, 7 1, 50)	93 93% (2, 7 67, 45)	94 43% (2, 8 26, 47)	91 62% (2, 6 82, 48)	93 57% (2, 5 37, 36)	92 83% (2, 6 24, 44)	92 57% (2, 6 98, 48)	95 19% (2, 5 49, 35)	94 86% (2, 6 99, 47)
Adeno Beer	88 33% (2, 12 12, 50)	88 39% (2, 13 53, 50)	86 96% (2, 10 62, 50)	87 64% (2, 10 64, 48)	87 83% (2, 12 52, 49)	87 29% (2, 12 85, 48)	88 38% (2, 13 64, 48)	88 62% (2, 16 07, 47)	88 52% (2, 14 72,48)

Table 14. Performance of subsets S with Recursive algorithms and their baselines, with ranking and selection on training sets.

	KNN				SVM-Soft				SVM-Hard			
	TNoM	Rec-TNoM	*f*-Test	Rec-*f*-Test	TNoM	Rec-TNoM	*f*-Test	Rec-*f*-Test	TNoM	Rec-TNoM	*f*-Test	Rec-*f*-Test
Beer	99 06%	99 27%	98 93%	99 38%	99 27%	99 27%	98 93%	99 27%	100%	98 97%	99 89%	99 48%
Small Beer	98 94%	98 98%	98 93%	99 50%	98 94%	98 98%	98 93%	99 29%	99 68%	99 08%	99 24%	99 90%
Squamous	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Gordon	99 45%	99 06%	99 17%	99 17%	99 11%	99 06%	99 23%	99 28%	99%	99 06%	99 28%	99 23%
Bhttacharjee	97 47%	97 61%	97 41%	97 91%	98 45%	97 41%	97 91%	98 04%	98 45%	97 66%	98 09%	97 98%
Golub	95 47%	96 11%	96 24%	96 80%	94 69%	95 04%	95 45%	95 49%	95 27%	94 85%	96 35%	96 70%
Alon	84 98%	82 76%	82 26%	79 45%	84 79%	79 81%	79 60%	79 21%	80 33%	77 64%	76 19%	77 24%
Adeno Beer	79 61%	79 12%	78 94%	78%	76 92%	79 19 %	76 81%	75 83%	77 70%	80 46%	76 43%	76 56%

Table 15. Performance of subsets Best-S with Recursive algorithms and their baselines, with ranking and selection on training sets.

	KNN				SVM-Soft				SVM-Hard			
	TNoM	Rec-TNoM	*f*-test	Rec-*f*-test	TNoM	Rec-TNoM	*f*-test	Rec-*f*-test	TNoM	Rec-TNoM	*f*-test	Rec-*f*-test
Beer	100% (1,1 43 26)	100% (1,1 18,2)	100% (1, 1 33, 12)	100% (1, 1 14,3)	100% (1 1 61,26)	100% (1,1 38,22)	100% (1, 1 45, 12)	100% (1, 1 18, 4)	100% (1,1 29,12)	100% (1,1 18,2)	100% (1,1 35,8)	100% (1,1 18,4)
Small Beer	99 89% (1 1 19 2)	100% (1 1 2 2)	98 93% (1,1,1)	100% (1, 1 37,8)	99 89% (1 1 19,2)	99 80% (1,1 18,2)	98 93% (1 1 1)	100% (1, 1 37, 8)	100% (1 1 19,2)	100% (1,1 2,2)	100% (1,1 75,34)	100% (1,1 39,12)
Squamous	100% (1 1 43 3)	100% (1,1 49,3)	100% (1, 1 04, 4)	100% (1, 1 03,4)	100% (1 1 27,2)	100% (1,1 27,2)	100% (1 1 04, 4)	100% (1, 1 03, 4)	100% (1,1 41,2)	100% (1,1 49,3)	100% (1,1 03 3)	100% (1 1 02,3)
Gordon	99 94% (1 4 76,48)	99 89% (1 6 19 29)	99 94% (1 3 88 16)	99 94% (1,5 27,35)	99 67% (1 3 90 45)	99 78% (1,5 32,24)	99 94% (1,4 28 34)	99 89% (1,4 24 17)	99 67% (1,4 25,50)	99 50% (1 6 22 28)	99 78% (1,4 25,17)	99 61% (1 4 85 40)
Bhttacharjee	99 37% (1,3 95,49)	99 30% (1 4 14 30)	98 69% (1 1 71 25)	98 89% (1, 1 8, 24)	98 82% (1,3 08,23)	98 30% (1 1 99 14)	98 76% (1 1 05 6)	98 82% (1 1 22,13)	99 30% (1,2 51,19)	99 03% (1 3 52 50)	99 16% (1,1 87,18)	99 16% (1,1 37,17)
Golub	98 71% (1 3 57 11)	98 92% (1,3 78,20)	98 77% (1,3 25,14)	98 65% (1 6 08 38)	98 12% (1 3 82 15)	98 82% (1,4 49,48)	98 65% (1,4 32,42)	97 65% (1 6 77, 48)	98 25% (1,4 09,34)	99 17% (1,4 92,28)	98 92% (1,4 94,41)	98 27% (1 6 95 48)
Alon	91 48% (1,4 70,47)	91 21% (1 7 70 44)	94 93% (1,3 77,18)	93 29% (1, 3 92 34)	92 93% (1,5 75,45)	91 74% (1,6 89,46)	93 17% (1,3 19,45)	91 98% (1 3 94, 46)	93 83% (1,6 28,28)	92 57% (1,5 74,39)	93 50% (1,4 87,23)	90 86% (1 3 07 46)
Adeno Beer	92 08% (1,9 18,45)	91 51% (1 11 29 50)	93 16% (1,9 24,43)	91 36% (1,11 73 50)	89 52% (1,10 66 50)	90 72% (1,11 62,47)	88 61% (1,8 90,47)	87 84% (1 10 26 49)	92 25% (1,9 18,45)	91 54% (1 11 21 46)	90 78% (1,9 96,36)	88 83% (1 12 14 49)

5.1. Reporting a single subset of genes

As already mentioned in train and test procedure since the feature ranking and selection are performed in each fold of the ten-fold-cross-validation, then different subsets *S* and *Best-S* are obtained after the ten iterations of the cross-validation process. So particularly for our last set of experiments (Tables 12 to 15), we used two frameworks for reporting/returning a *single* gene subset (*S* or *Best-S*) out of the hundred such subsets we obtained after the ten runs of ten-fold cross- validation.

5.1.1. Frequent Genes Reporting

In the first framework we report the genes, which appear most often in all hundred cross-validation folds; that is after 10 runs of 10 fold cross validation we reach 100 different subsets of genes; we created an array, whose size is the number of genes (with the initial frequency values of zero) and each time that a gene is selected, we increase its frequency by one and finally we report the 50 most frequent genes.

We have done this set of reporting with 8 datasets of [1] with exactly the same pre-processing steps; here after obtaining the frequent genes in the framework of 10 runs of 10 fold cross validation, we applied Leave One Out Cross Validation on the returned subset of genes with classifiers and Tables 16 to 20 show the performances of Best-*S* and *S* with different classifiers. In addition, we compared the performance of the recursive algorithm implemented with different ranking criteria with their baselines (See Tables 19 and 20). By comparing the results achieved by the pair based selection approaches, we

can see the performances achieved by our *graph-based* approaches are better than simple LSGP. In addition, if we compare the performances obtained by our recursive algorithm with the baselines, significant improvement for those datasets, that there is enough room to improve from the base lines to our recursive algorithm, is observed. However for those datasets that baselines return accuracies near to 100%, there is not enough room to improve anymore and the improvement of our recursive algorithm is trivial.

Table 16. Performance of frequent subsets of genes of LSGP approach

	KNN		DLD		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (4)	100%	100% (8)	100%	100% (4)	100%	100% (4)	100%
Small Beer	100% (3)	100%	98 96% (12)	98 96%	100% (3)	100%	100% (3)	100%
Adeno Beer	89 53% (27)	81 40%	91 86% (13)	90 70%	90 70% (9)	86 05%	89 53% (37)	83 72%
Golub	100% (27)	100%	98 61% (14)	98 61%	100% (27)	98 61%	100% (28)	98 61%
Bhttacharjee	99 36% (11)	99 36%	99 36% (7)	99 36%	99 36% (2)	99 36%	99 36% (10)	99 36%
Gordon	100% (31)	100%	98 90% (7)	98 90%	100% (31)	100%	100% (30)	100%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	85 48% (4)	83 87%	91 94% (9)	91 94%	90 32% (4)	85 48%	91 94% (15)	85 48%

Table 17 Performance of frequent subsets of genes of DF-LSGP approach

	KNN		DLD		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (5)	100%	100% (5)	100%	100% (5)	100%	100% (5)	100%
Small Beer	100% (5)	98 96%	98 96% (4)	97 92%	100% (2)	100%	100% (5)	100%
Adeno Beer	88 37% (31)	88 37%	90 70% (11)	90 70%	90 70% (9)	84 88%	87 21% (42)	87 21%
Golub	100% (4)	100%	100% (4)	100%	100% (7)	100%	100% (4)	100%
Bhttacharjee	100% (3)	100%	100% (32)	99 36%	100% (2)	100%	100% (3)	100%
Gordon	100% (2)	100%	99 45% (31)	98 90%	100% (10)	100%	100% (9)	100%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	91 94% (6)	85 48%	91 94% (12)	91 94%	91 94% (4)	87 10%	90 32% (5)	85 48%

Table 18 Performance of frequent subsets of genes of BF-LSGP approach

	KNN		DLD		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (5)	100%	100% (5)	100%	100% (5)	100%	100% (5)	100%
Small Beer	100% (5)	98 96%	98 96% (4)	98 96%	100% (2)	98 96%	100% (5)	98 96%
Adeno Beer	87 21% (49)	87 21%	90 70% (42)	90 70%	91 86% (34)	86 05%	88 37% (49)	88 37%
Golub	100% (3)	100%	100% (3)	100%	100% (2)	100%	100% (3)	100%
Bhttacharjee	100% (2)	100%	99 36% (30)	97 44%	100% (3)	100%	100% (3)	99 36%
Gordon	100% (3)	100%	98 90% (35)	98 90%	100% (3)	99 45%	100% (3)	99 45%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	93 55% (2)	82 26%	91 94% (13)	91 94%	93 55% (4)	91 94%	93 55% (4)	87 10%

Table 19 Performance of the frequent subsets of genes with TNoM vs. Rec-TNoM

	SVM-Soft[TNoM]		SVM-Soft[Rec-TNoM]		KNN[TNoM]		KNN[Rec-TNoM]		DLD[TNoM]		DLD[Rec-TNoM]		SVM-Hard[TNoM]		SVM-Hard [Rec-TNoM]	
	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*
Beer	100% (3)	100%	100% (3)	98 96%	100% (3)	100%	100% (3)	98 96%	100% (4)	100%	100% (6)	100%	100% (3)	100%	100% (3)	98 96%
Small Beer	100% (3)	100%	100% (3)	98 96%	100% (3)	98 96%	100% (3)	98 96%	98 96% (14)	98 96%	98 96% (11)	98 96%	100% (2)	100%	100% (2)	98 96%
Squamous	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (1)	100%	100% (1)	100%	100% (2)	100%	100% (2)	100 %
Gordon	100% (8)	98 90%	99 45% (7)	98 90%	100% (45)	99 45%	100% (44)	99 45%	98 90% (8)	98 90%	98 90% (6)	98 90%	100% (9)	98 90%	99 45% (19)	98 34%
Bhattacharjee	98 72% (1)	98 08%	98 72% (1)	98 72%	98 08% (5)	98 08%	99 36% (10)	99 36%	98 72% (2)	98 08%	99 36% (9)	98 72%	98 72% (12)	98 08%	98 72% (15)	98 08%
Golub	98 61% (6)	95 83%	100% (5)	95 83%	98 61% (7)	97 22%	98 61% (13)	97 22%	98 61% (6)	98 61%	100% (5)	98 61%	100% (12)	97 22%	100% (7)	98 61%
Alon	90 32% (6)	87 10%	91 94% (4)	91 94%	88 71% (12)	85 48%	88 71% (10)	85 48%	91 94% (9)	87 10%	91 94% (11)	91 94%	90 32% (10)	88 71%	90 32% (3)	82 26%
Adeno Beer	94 19% (23)	88 37%	95 35% (43)	86 05%	93 02% (24)	87 21%	94 19% (42)	88 37%	88 37% (37)	84 88%	94 19% (35)	94 19%	96 51% (24)	84 88%	98 84% (34)	90 70%

Table 20. Performance of the frequent subsets of genes with *f-test* vs. Rec-*f*-test

	SVM-Soft[*f-test*]		SVM-Soft[Rec-*f-test*]		KNN[*f-test*]		KNN[Rec-*f-test*]		DLD[*f-test*]		DLD[Rec-*f-test*]		SVM-Hard[*f-test*]		SVM-Hard[Rec-*f-test*]	
	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*	Best-*S*	*S*
Beer	100% (4)	98 96%	100% (3)	100%	100% (3)	98 96%	100% (3)	100%	100% (5)	100%	100% (8)	100%	100% (3)	100%	100% (3)	100%
Small Beer	100% (4)	98 96%	100% (4)	100%	100% (3)	98 96%	100% (6)	100%	100% (5)	98 96%	100% (9)	98 96%	100% (3)	100%	100% (2)	100%
Squamous	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%
Gordon	100% (7)	98 90%	100% (10)	100%	99 45% (6)	99 45%	100% (7)	99 45%	100% (10)	99 45%	100% (42)	100%	100% (10)	99 45%	100% (10)	100%
Bhattacharjee	98 72% (1)	98 08%	99 36% (39)	99 36%	98 08% (4)	98 08%	99 36% (21)	98 72%	99 36% (32)	98 72%	99 36% (5)	99 36%	98 08% (1)	98 08%	99 36% (45)	99 36%
Golub	97 22% (7)	93 06%	98 61% (22)	98 61%	98 61% (15)	97 22%	100% (32)	100%	100% (7)	98 61%	100% (24)	100%	100% (7)	97 22%	100% (26)	100%
Alon	90 32% (5)	79 03%	91 94% (45)	90 32%	87 10% (20)	77 42%	87 10% (12)	83 87%	91 94% (12)	88 71%	91 94% (17)	88 71%	95 16% (10)	82 26%	91 94% (50)	91 94%
Adeno Beer	89 53% (14)	83 72%	97 67% (24)	94 19%	91 86% (33)	82 56%	93 02% (9)	89 53%	89 53% (32)	89 53%	91 86% (11)	90 70%	87 21% (10)	81 40%	96 51% (25)	95 35%

The reported genes for Alon and Golub datasets with our BF-LSGP approach, which gives us better performance in comparison to our other *pair-based* selection approaches in this framework, are listed in index 1. We, also, listed the genes reported by our Rec-*f-test* (see tables 30 to 33).

5.1.2. Best Subset Report

In the second framework for returning a single subset of genes called *"Best Subset"*, each time after obtaining a subset of gene in each fold of *m*-fold cross validation, we applied Leave One Out Cross Validation based on that subset of genes, whose size is 50, on whole of the samples. Thus after 10 runs of 10 fold cross validation among 100 different subsets, we report the subset giving us the best performance. For instance, for XX-classifier we report the subset of 50 genes whose performance was the highest one among these 100 subsets with XX-classifier. After obtaining that subset of gene we report the accuracies of Best-*S* and *S* based on Leave One out Cross Validation. Tables 21 to 25 show the comparison of Best-S and S between our different approaches with SVM-Soft, KNN and SVM-Hard classifiers. By looking at the tables given, we will see that the pair based selection approaches are completely comparable with each other and for this set of reporting genes. We also compared the performance of our recursive algorithm implemented with two ranking criteria with their base lines (See Tables 24 and 25). In this set of experiment the recursive algorithm again dramatically outperforms the baselines.

Table 21 Performance of best subsets of genes of LSGP approach

	KNN		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (4)	100%	100% (4)	100%	100% (4)	100%
Small Beer	100% (21)	100%	100% (13)	100%	100% (5)	100%
Adeno Beer	91 86% (47)	91 86%	94 19% (31)	89 53%	91 86% (48)	91 86%
Golub	100% (4)	100%	100% (4)	100%	100% (12)	100%
Bhttacharjee	100% (3)	99 36%	100% (2)	99 36%	100% (3)	99 36%
Gordon	100% (4)	100%	100% (3)	100%	100% (2)	100%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	93 55% (2)	91 94%	91 94% (5)	91 94%	93 55% (6)	93 55%

Table 22 Performance of best subsets of genes of DF-LSGP approach

	KNN		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (4)	100%	100% (4)	100%	100% (4)	100%
Small Beer	100% (4)	100%	100% (5)	100%	100% (4)	100%
Adeno Beer	90 70% (5)	90 70%	91 86% (49)	91 86%	91 86% (46)	88 37%
Golub	100% (3)	100%	100% (11)	100%	100% (4)	100%
Bhttacharjee	100% (5)	100%	100% (4)	100%	100% (3)	100%
Gordon	100% (5)	100%	100% (19)	100%	100% (16)	100%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	93 55% (3)	91 94%	96 77% (18)	96 77%	95 16% (45)	95 16%

Table 23 Performance of best subsets of genes of BF-LSGP approach

	KNN		SVM-Soft		SVM-Hard	
	Best-*S*	***S***	**Best-*S***	***S***	**Best-*S***	***S***
Beer	100% (4)	100%	100% (4)	100%	100% (4)	100%
Small Beer	100% (5)	100%	100% (5)	100%	100% (5)	100%
Adeno Beer	90 70% (49)	90 70%	91 86% (27)	90 70%	89 53% (50)	89 53%
Golub	100% (2)	100%	100% (5)	100%	100% (2)	100%
Bhttacharjee	100% (3)	100%	100% (2)	100%	100% (5)	100%
Gordon	100% (13)	100%	100% (3)	100%	100% (11)	100%
Squampus	100% (1)	100%	100% (1)	100%	100% (1)	100%
Alon	95 16% (41)	91 94%	95 16% (4)	95 16%	95 16% (31)	95 16%

Table 24Performance of the best subsets of genes with *f-test* vs. Rec-*f-test*

	SVM-Soft[*f*-test]		SVM-Soft[Rec-*f*-test]		KNN[*f*-test]		KNN[Rec-*f*-test]		SVM-Hard[*f*-test]		SVM-Hard[Rec-*f*-test]	
	S	**Best-*S***	**S**	**Best-*S***	**S**	**Best-*S***	***S***	**Best-*S***	**S**	**Best-*S***	**S**	**Best-*S***
Beer	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)
Small Beer	98 96%	100% (46)	100%	100% (3)	98 96%	100% (4)	100%	100% (3)	100%	100% (1)	100%	100% (1)
Squamous	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)	100%	100% (1)
Gordon	100%	100% (7)	100%	100% (6)	100%	100% (6)	100%	100% (6)	100% (6)	100%	100% (7)	100%
Bhattacharjee	99 36%	99 36% (45)	99 36%	99 36% (11)	98 72%	99 36% (23)	99 36%	99,36% (4)	98 72%	98 72% (50)	99 36% (6)	99 36%
Golub	97 22%	97 22% (5)	100%	100% (38)	100%	100% (48)	100%	100% (40)	100%	100% (4)	100%	100% (7)
Alon	91 94%	91 94% (49)	93 55%	95 16% (5)	90 32%	90 32% (49)	91 94%	95 16% (29)	88 71%	88 71% (23)	95 16%	96 77% (47)
Adeno Beer	90 70%	91 86% (49)	98 84%	100% (41)	93 02%	94 19% (30)	97 67%	97 67% (45)	89 53%	94 19%	98 84%	100% (47)

Table 25 Performance of the best subsets of genes with TNoM vs Rec-TNoM

	SVM Soft[TNoM]		SVM-Soft[Rec-TNoM]		KNN[TNoM]		KNN[Rec-TNoM]		SVM-Hard[TNoM]		SVM-Hard[Rec-TNoM]	
	S	**Best-*S***	**S**	**Best-*S***	**S**	**Best-*S***	***S***	**Best-*S***	***S***	**Best-*S***	***S***	**Best-*S***
Beer	100%	100% (4)	100%	100% (8)	100%	100% (3)	100%	100% (2)	100%	100% (3)	100%	100% (3)
Small Beer	100%	100% (4)	100%	100% (4)	100%	100% (3)	100%	100% (3)	100%	100% (1)	100%	100% (1)
Squamous	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)	100%	100% (2)
Gordon	100%	100% (50)	100%	100% (11)	100%	100% (45)	100%	100% (5)	100%	100% (50)	100%	100% (13)
Bhattacharjee	98 72%	99 36% (5)	99 36%	99 36% (8)	98 72%	99 36% (3)	99 36%	99 36% (2)	98 72%	99 36% (11)	99 36%	99 36% (27)
Golub	97 22%	98 61% (16)	100%	100% (11)	98 61%	98 61% (7)	100%	100% (12)	100%	100% (50)	100%	100% (11)
Alon	91 94%	91 94% (50)	95 16%	95 16% (30)	91 94%	91 94% (41)	91 94%	91 94% (20)	91 94%	91 94% (6)	95 16	95 16% (48)
Adeno Beer	94 19%	94 19% (36)	97 67%	97 67% (24)	93 02%	94 19% (39)	94 19%	95 35% (23)	95 35%	95 35% (50)	97 67%	97 67% (35)

The reported subsets of genes for Alon and Golub datasets of our DF-LSGP approach, which gives us better performance with SVM-Soft Classifier in comparison to our other pair based approaches in the framework of best subset report of genes, are listed in index 1 (See tables 34 and 35). We, also, listed the subsets of genes for Alon and Golub datasets, which give us better performance by SVM-Soft Classifier in this framework with Rec-TNoM (see tables 36 to 37).

6. Summary

Table 26 shows a summary of attributes of different algorithms introduced in this research. We have compared algorithms based on whether they select pairs of genes of single genes, graph based approach, Linear Separability and finally whether the continuous data is used or discrete data. Table 27, also, shows the best accuracies achieved with different classifiers used in this research with different approaches introduced in this research on whole datasets.

Table 26. Summary of attributes of algorithms

	LS	LSGP	DF-LSGP	BF-LSGP	Recursive	MIQ (mRMR)	Greedy Pair Method Of [2]
Pair Selection	X	X	X	X			X
Graph Based			X	X			
LS	X	X	X	X			
Continuous Data	X	X	X	X	X		X
Discrete Data						X	

Table 27. Summary of the best accuracies achieved with different introduced approaches

	LSGP	DF-LSGP	BF-LSGP	Rec-ftest	Rec-TNOM
Beer	100%	100%	100%	100%	100%
	(4)	(4)	(4)	(2)	(3)
Small Beer	100%	100%	100%	100%	100%
	(4)	(4)	(4)	(3)	(3)
Squamous	100	100	100	100	100
	(1)	(1)	(1)	(1)	(2)
Gordon	100%	100%	100%	100%	100%
	(2)	(2)	(2)	(5)	(5)
Bhattacharjee	100%	100%	100%	99.36%	99.36%
	(2)	(2)	(2)	(6)	(3)
Golub3	100%	100%	100%	100%	98.61%
	(4)	(47)	(20)	(12)	(15)
Alon3	96.77%	96.77%	96.77%	95.16%	96.77%
	(10)	(11)	(11)	(19)	(33)
Adeno Beer	90.70%	90.70%	90.70%	96.51%	96.51%
	(34)	(18)	(18)	(19)	(15)

7. Comparison of running time

The running time of the gene selection algorithms depends on the number of genes and number of training samples. We have compared the running times of the graph-based approach and recursive algorithm implemented with *f-test*. Table 28 shows the running time for these two algorithms for all of the datasets used in this research. The running times were recorded using MATLAB codes on Intel CPU with 3.19 GHz processor. Table 28 shows that the recursive algorithm is much faster than the graph-based method of pair selection, for all datasets except for squamous dataset, which is considered as a *very high-separable* dataset and has many LS-genes, so the graph based

algorithm finds all of the 50 genes with LS-genes and does not go further to find LS-pairs of genes.

Table 28. Comparison of running times of pair selection and the recursive lagorithm

	Rec|*f-test*| (Seconds)	**DF-LSGP** (Seconds)	**Degree of Separability**	**Nb of genes**	**Nb of Samples**
Beer	73 38	1126 24	High	7129	96
Small Beer	50 10	644 80	Very High	4966	96
Squamous	42 87	0 09	Very High	4295	41
Gordon	136 57	2631 05	Very High	12533	181
Bhttacharjee	44 04	367 83	Very High	4392	156
Golub 3	72 59	761 30	High	7129	72
Alon 3	19 72	836 36	Border Line	2000	62
Adeno Beer	50 26	8098 88	No	4966	86

Also, we see the running time of the graph based algorithm depends on the degree of sperabaility of dataset; as an illustration, for Adeno Beer which is considered as a *non-separable* dataset, we have to delete Support Vectors with largest Lagrange coefficients and repeat the procedure to reach LS-features. In addition, the number of samples and genes influence the speed of both recursive algorithm and graph-based of pair selection. For instance, we see that the running time of Gordon dataset is considered high in comparison to other datasets due to large number of genes and samples.

CHAPTER V

DATASETS AND MATERIALS

1. Datasets

To evaluate the performance of the proposed methods, we have done extensive experiments on eight publicly available microarray gene expression datasets, namely, Golub [4], Alon [5], Gordon[10], Beer [7], Small Beer [7], AdenoBeer [7], Bhattacharjee [8] and Squamous [8] datasets shown in table 29.

Table 29. Gene Expression Datasets used

Dataset Name	Cancer Type	Nb of Genes	Nb of Samples	# of samples of Class 1	# samples of Class 2	Degree of Separability
Beer	Lung	7129	96	86	10	High
Small Beer	Lung	4966	96	86	10	Very High
Squamous	Lung	4295	41	21	20	Very High
Gordon	Lung	12533	181	150	31	Very High
Bhattacharjee	Lung	4392	156	139	17	Very High
Golub	Leukemia	7129	72	47	25	High
Alon	Colon	2000	62	40	22	Border Line
Adeno Beer	Lung	4966	86	67	19	No

2. Pre-Processing Steps

For datasets we did the following preprocessing steps in the same manner of [1]:

- Trimming: all values lower than 100 were set to 100, and all values higher than 16,000 were set to 16,000.

- Natural logarithm: The Natural logarithm (*ln(x)*) was taken for each value.
- Standardizing: Each sample was standardized to have a mean of 0 and a standard deviation of 1.

Additional preprocessing steps are as follows:

- Small Beer Dataset is a subset of Beer dataset; it contains the same 96 samples but only those 4,966 genes used by Beer et al. in the original paper [7]
- AdenoBeer is a subset of the Small Beer dataset. It contains only the 86 lung ADCA tumors, divided into two classes of 67 stage-1 and 19 stage-3 tumors.
- Bhattacharjee: This data set contains five classes, among these five classes we chose two classes of 139 lung-cancer ADCAs Versus the 17 normal tissues, totaling 156 samples. In the original Dataset, expression levels are given for 12,600 genes. However of the values are outside the range 100-16,000; thus after trimming the values, many artifact-existence of many millions of separate pairs. To avoid this, we applied a variation filter: only gene showing two fold variation and a gap of at least 50 between the minimal and maximal values (across the 156 samples) were taken. This process left us with 4,392 genes.
- Squamous: This dataset is based on the original Bhattacharjee dataset. It contains 21 squamous cell lung carcinoma tumors and 20 pulmonary carcinoid tumors, a total of 41 samples. The sample variation filter described for Bhattacharjee was applied here, leaving us 4,295 genes.

For two other dataset called Golub2 and Alon2 we did the same preprocessing steps, done in [2], in order to have a sound comparison between Gene Subset returned by our approach and theirs. The preprocessing for these two datasets is as follows:

- Logarithmic transformation: Base 10 logarithmic transformation
- Standardizing: For each gene, subtract the mean and divide by standard deviation.

For Golub2 the following additional preprocessing step is done (Similar to [2]); thresholding with a floor of 1 and filtering by excluding genes with $max/min \leq 500$. This leaves us with a dataset of 3,934 genes.

For Alon3 and Golub3, we pre-processed them similar to [6], to have genes with mean of 0 and standard deviation of 1.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

1. Conclusion and Discussion

In this research in the pair based selection algorithms we investigated the idea of using the concept of linear separability of gene expression data for the purpose of gene subset selection. We showed that the Containment Angle (CA) can be used to rank linearly separating pairs of genes. We, also, introduced a new ranking criterion for ranking LS-genes. We proposed different gene subset selection methods, LS, LSGP, DF-LSGP and BF-LSGP, which select linearly separating features using our ranking criteria. Extensive experiments are carried out showing that our approaches are at least comparable to current filtering methods, which are based on selecting gene-pairs rather than only single genes.

However in univariate filter methods and even pair based selection we still have the problem of redundancy; that is, when a pair of gene is selected some of these samples may cause non-linear separability and may never be classified correctly by adding new pairs of genes and we keep adding redundant genes covering only some parts of the space; hence the returned subset of genes may never cover the space perfectly; this one was our motivation to study the effect of samples' selection for ranking and selecting genes to overcome the problem of redundancy; a new recursive feature subset selection algorithm, emphasizing on linear separation between samples has been introduced. Furthermore, we proved that not all of the samples are important for ranking genes. Our

algorithm, which is easy to implement, only considers those samples causing non Linear Separability, for ranking and selecting gens. It, also, covers the space better and broadly. In this thesis we carried out extensive experiments by two different ranking criteria called *f-test* and TNoM; we, also, compared the performance of our recursive algorithm with a well-known algorithm in the field of feature selection and the baselines. The extensive experiments on benchmark cancer classification datasets substantiated that our recursive algorithm yields much better results than their baselines.

2. Future Work

As a future research, we plan to generalize the theorem of [1] for generating all linearly separating t-tuples $g_{I\ t} = (g_{i1}, g_{i2}, \ldots, g_{it})$ from a given data set and for a given size $t \geq 3$. Another interesting extension we can have is to study wrapper methods based on selecting (not necessarily linearly separating) gene-pairs. In this regard, our graph-based methods, DF-LSGP and BF-LSGP, will be modified to back-track or continue the search depending on the classifier's error on the current subset. In this thesis we devised ranking criteria applied only to LS-features, which is quite restrictive. Hence, we are devising general ranking criteria which will apply to *all* features, and in such a way that LS-features are ranked very high.

Moreover, for our recursive algorithm it is interesting to take into consideration ranking and selecting pairs of genes or n-tuple of genes instead of individual genes. Also, currently we are testing the performance of our recursive algorithm with appropriate ranking criteria on discretized data.

APPENDICES

Gene Report

Table 30. Top 50 frequent genes for Golub Dataset using BF-LSGP approach

Rank	Gene ID	Gene annotation
1	M31523_at	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
2	M23197_at	CD33 CD33 antigen (differentiation antigen)
3	M55150_at	FAH Fumarylacetoacetate
4	L07633_at	INTERFERON GAMMA UP-REGULATED I-5111 PROTEIN PRECURSOR
5	M14016_at	UROD Uroporphyrinogen decarboxylase
6	M31166_at	PTX3 Pentaxin-related gene, rapidly induced by IL-1 beta
7	X85116_rna1_s_at	Epb72 gene exon 1
8	M30703_s_at	Amphiregulin (AR) gene
9	L12168_at	ADENYLYL CYCLASE-ASSOCIATED PROTEIN 1
10	AFFX-HUMTFRR/M11507_M_at	AFFX-HUMTFRR/M11507_M_at (endogenous control)
11	D86976_at	KIAA0223 gene, partial cds
12	U82759_at	GB DEF = Homeodomain protein HoxA9 mRNA
13	M60974_s_at	DDIT1 DNA-damage-inducible transcript 1
14	M15205_at	TK1 Thymidine kinase 1, soluble
15	D84145_at	WS-3 mRNA
16	D88270_at	GB DEF = (lambda) DNA for immunoglobin light chain
17	U05259_rna1_at	MB-1 gene
18	X59350_at	CD22 CD22 antigen
19	U06452_at	MLANA Differentiation antigen melan-A
20	Y12670_at	LEPR Leptin receptor
21	HG2724-HT2820_at	Oncogene Tls/Chop, Fusion Activated
22	M31303_rna1_at	Oncoprotein 18 (Op18) gene
23	Z14982_rna1_at	MHC-encoded proteasome subunit gene LAMP7-E1 gene (proteasome subunit LMP7) extracted from
24	M81933_at	CDC25A Cell division cycle 25A
25	AFFX-HUMTFRR/M11507_3_at	AFFX-HUMTFRR/M11507_3_at (endogenous control)
26	X66401_cds1_at	LMP2 gene extracted from H sapiens genes TAP1, TAP2, LMP2, LMP7 and DOB
27	HG1612-HT1612_at	Macmarcks
28	U37122_at	ADD3 Adducin 3 (gamma)
29	M95678_at	PLCB2 Phospholipase C, beta 2
30	M29610_s_at	GYPE Glycophorin E
31	U46751_at	Phosphotyrosine independent ligand p62 for the Lck SH2 domain mRNA
32	M89957_at	IGB Immunoglobulin-associated beta (B29)
33	J02982_f_at	GYPB Glycophorin B
34	M27891_at	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
35	U02081_at	Guanine nucleotide regulatory protein (NET1) mRNA
36	U09770_at	Cysteine-rich heart protein (hCRHP) mRNA
37	M28713_at	NADH-CYTOCHROME B5 REDUCTASE
38	X62654_rna1_at	ME491 gene extracted from H sapiens gene for Me491/CD63 antigen
39	X95735_at	Zyxin
40	M64231_rna1_at	Spermidine synthase gene
41	M92287_at	CCND3 Cyclin D3
42	L09209_s_at	APLP2 Amyloid beta (A4) precursor-like protein 2
43	U40343_at	CDK inhibitor p19INK4d mRNA
44	D42043_at	KIAA0084 gene, partial cds
45	X03934_at	GB DEF = T-cell antigen receptor gene T3-delta
46	X71973_at	GPX4 Phospholipid hydroperoxide glutathione peroxidase
47	X51521_at	VIL2 Villin 2 (ezrin)
48	J05243_at	SPTAN1 Spectrin, alpha, non-erythrocytic 1 (alpha-fodrin)
49	M63138_at	CTSD Cathepsin D (lysosomal aspartyl protease)
50	X82240_rna1_at	TCL1 gene (T cell leukemia) extracted from H sapiens mRNA for Tcell leukemia/lymphoma 1

Table 31. Top 50 frequent genes for Alon Dataset using BF-LSGP approach

Rank	Gene ID	Gene annotation
1	Z50753	H sapiens mRNA for GCAP-II/uroguanylin precursor
2	H22579	INTEGRIN ALPHA 6 PRECURSOR (Homo sapiens)
3	R87126	MYOSIN HEAVY CHAIN NONMUSCLE (Gallus gallus)
4	H29546	NEUROTENSIN RECEPTOR (Homo sapiens)
5	T94993	FIBROBLAST GROWTH FACTOR RECEPTOR 2 PRECURSOR (Homo sapiens)
6	X66975	H sapiens mRNA for heterogeneous nuclear ribonucleoprotein
7	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds
8	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
9	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
10	R70939	TRANSCRIPTION FACTOR TAU 131 KD SUBUNIT (Saccharomyces cerevisiae)
11	X14958	Human hmgI mRNA for high mobility group protein Y
12	X12548	Human mRNA for lysosomal acid phosphatase (EC 3 1 3 2)
13	H20709	MYOSIN LIGHT CHAIN ALKALI, SMOOTH-MUSCLE ISOFORM (HUMAN),
14	M63391	Human desmin gene, complete cds
15	T51023	HEAT SHOCK PROTEIN HSP 90 BETA (HUMAN)
16	R48303	TYROSINE RICH ACIDIC MATRIX PROTEIN (Bos taurus)
17	R36977	P03001 TRANSCRIPTION FACTOR IIIA ,
18	R44301	MINERALOCORTICOID RECEPTOR (Homo sapiens)
19	M93010	Human epithelial cell marker protein 1 (HMe1) mRNA, complete cds
20	H49870	MAD PROTEIN (Homo sapiens)
21	D15049	Human mRNA for protein tyrosine phosphatase
22	R90908	PUTATIVE SERINE/THREONINE-PROTEIN KINASE T17E9 1 IN CHROMOSOME III (Caenorhabditis elegans)
23	X70297	NEURONAL ACETYLCHOLINE RECEPTOR PROTEIN, ALPHA 7 CHAIN (HUMAN),
24	L40904	H sapiens peroxisome proliferator activated receptor gamma, complete cds
25	L13385	Homo sapiens(clone 71) Miller-Dieker lissencephaly protein (LIS1) mRNA, complete cds
26	R74208	GENERAL NEGATIVE REGULATOR OF TRANSCRIPTION SUBUNIT 4 (Saccharomyces cerevisiae)
27	T63484	Human ornithine decarboxylase antizyme (Oaz) mRNA, complete cds
28	X80692	H sapiens ERK3 mRNA
29	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
30	T51571	P24480 CALGIZZARIN
31	H77597	H sapiens mRNA for metallothionein (HUMAN),
32	M94132	Human mucin 2 (MUC2) mRNA sequence
33	U14631	Human 11 beta hydroxysteroid dehydrogenase type II mRNA, complete cds
34	X16504	Human ENO3 mRNA for beta enolase (EC 4 2 1 11)
35	M22382	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN),
36	M16827	Human medium chain acyl CoA dehydrogenase (ACADM) mRNA, complete cds
37	X16354	Human mRNA for transmembrane carcinoembryonic antigen BGPa (formerly TM1-CEA)
38	R62459	TROPONIN C, ISOFORM 2 (Balanus nubilis)
39	L00352	Human low density lipoprotein receptor gene, exon 18
40	H64489	LEUKOCYTE ANTIGEN CD37 (Homo sapiens)
41	J02854	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN),contains element TAR1
42	H02630	TRANSCRIPTIONAL REPRESSOR PROTEIN YY1 (Homo sapiens)
43	T71025	Human (HUMAN),
44	R61359	BASIGIN PRECURSOR (Gallus gallus)
45	D14812	Human mRNA for ORF, complete cds
46	R08183	Q04984 10 KD HEAT SHOCK PROTEIN, MITOCHONDRIAL ,
47	M94203	Homo sapiens protein kinase gene, 3' end of cds and trinucleotide repeat region
48	R65697	ATP SYNTHASE A CHAIN (Trypanosoma brucei brucei)
49	R55310	S36390 MITOCHONDRIAL PROCESSING PEPTIDASE ,
50	H51196	INSULIN RECEPTOR RELATED RECEPTOR PRECURSOR (Cavia porcellus)

Table 32. Top 50 frequent genes for Golub Dataset using Rec-f-test approach

Rank	Gene ID	Gene annotation
1	D88422_at	CYSTATIN A
2	HG1612-HT1612_at	Macmarcks
3	J05243_at	SPTAN1 Spectrin, alpha, non erythrocytic 1 (alpha-fodrin)
4	M11722_at	Terminal transferase mRNA
5	M16038_at	LYN V-yes-1 Yamaguchi sarcoma viral related oncogene homolog
6	M23197_at	CD33 CD33 antigen (differentiation antigen)
7	M27891_at	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
8	M63138_at	CTSD Cathepsin D (lysosomal aspartyl protease)
9	M84526_at	DF D component of complement (adipsin)
10	M92287_at	CCND3 Cyclin D3
11	U46499_at	GLUTATHIONE S-TRANSFERASE, MICROSOMAL
12	X95735_at	Zyxin
13	L09209_s_at	APLP2 Amyloid beta (A4) precursor-like protein 2
14	M31523_at	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
15	Z15115_at	TOP2B Topoisomerase (DNA) II beta (180kD)
16	M31211_s_at	MYL1 Myosin light chain (alkali)
17	M83652_s_at	PFC Properdin P factor, complement
18	D88270_at	GB DEF = (lambda) DNA for immunoglobin light chain
19	X62654_rna1_at	ME491 gene extracted from H sapiens gene for Me491/CD63 antigen
20	X61587_at	ARHG Ras homolog gene family, member G (rho G)
21	U05259_rna1_at	MB-1 gene
22	M13690_s_at	C1NH Complement component 1 inhibitor (angioedema, hereditary)
23	Z78285_f_at	GB DEF = mRNA (clone 1A7)
24	X03934_at	GB DEF = T-cell antigen receptor gene T3-delta
25	M19507_at	MPO Myeloperoxidase
26	U09578_at	MAPKAP kinase (3pK) mRNA
27	X51521_at	VIL2 Villin 2 (ezrin)
28	X59417_at	PROTEASOME IOTA CHAIN
29	M55150_at	FAH Fumarylacetoacetate
30	U31248_at	ZNF174 Zinc finger protein 174
31	M22324_at	ANPEP Alanyl (membrane) aminopeptidase (aminopeptidase N, aminopeptidase M, microsomal
32	Y00339_s_at	CA2 Carbonic anhydrase II
33	L47738_at	Inducible protein mRNA
34	M16276_at	HLA-DQB1 Major histocompatibility complex, class II, DQ beta 1
35	M22960_at	PPGB Protective protein for beta-galactosidase (galactosialidosis)
36	M96995_s_at	GRB2 Growth factor receptor-bound protein 2
37	M89957_at	IGB Immunoglobulin-associated beta (B29)
38	M71243_f_at	GB DEF = Glycophorin Sta (type A) exons 3 and 4, partial
39	U35451_at	Heterochromatin protein p25 mRNA
40	X07743_at	PLECKSTRIN
41	M29696_at	IL7R Interleukin 7 receptor
42	U89922_s_at	LTB Lymphotoxin-beta
43	X52947_at	GJA1 Cardiac gap junction protein
44	M58286_s_at	TNFR1 Tumor necrosis factor receptor 1 (55kD)
45	AB000449_at	VRK1
46	M63379_at	CLU Clusterin (complement lysis inhibitor, testosterone-repressed prostate message 2, apolipoprotein J)
47	HG2981-HT3127_s_at	Epican, Alt Splice 11
48	D42043_at	KIAA0084 gene, partial cds
49	M93056_at	LEUKOCYTE ELASTASE INHIBITOR
50	U82759_at	GB DEF = Homeodomain protein HoxA9 mRNA

Table 33 Top 50 frequent genes for Alon Dataset using Rec-f-test approach

Rank	Gene ID	Gene annotation
1	M63391	Human desmin gene, complete cds
2	Z50753	H sapiens mRNA for GCAP-II/uroguanylin precursor
3	R87126	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
4	M22382	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN),
5	X12671	Human gene for heterogeneous nuclear ribonucleoprotein (hnRNP) core protein A1
6	J02854	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN),contains element TAR1
7	R36977	P03001 TRANSCRIPTION FACTOR IIIA ,
8	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds
9	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
10	H43887	COMPLEMENT FACTOR D PRECURSOR (Homo sapiens)
11	D25217	Human mRNA (KIAA0027) for ORF, partial cds
12	T86473	NUCLEOSIDE DIPHOSPHATE KINASE A (HUMAN),
13	T47377	S 100P PROTEIN (HUMAN)
14	X12369	TROPOMYOSIN ALPHA CHAIN, SMOOTH MUSCLE (HUMAN),
15	R44418	EBNA 2 NUCLEAR PROTEIN (Epstein barr virus)
16	H77597	H sapiens mRNA for metallothionein (HUMAN),
17	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
18	T49647	MYRISTOYLATED ALANINE-RICH C-KINASE SUBSTRATE (Homo sapiens)
19	M26697	Human nucleolar protein (B23) mRNA, complete cds
20	R67343	IMMEDIATE EARLY REGULATORY PROTEIN IE N (Autographa californica nuclear polyhedrosis virus)
21	T51023	HEAT SHOCK PROTEIN HSP 90 BETA (HUMAN)
22	Z48541	H sapiens mRNA for protein tyrosine phosphatase
23	D31885	Human mRNA (KIAA0069) for ORF (novel proetin), partial cds
24	H22579	INTEGRIN ALPHA 6 PRECURSOR (Homo sapiens)
25	M36634	Human vasoactive intestinal peptide (VIP) mRNA, complete cds
26	M28129	Homo sapiens eosinophil-derived neurotoxin (EDN) mRNA, complete cds
27	R15447	CALNEXIN PRECURSOR (Homo sapiens)
28	X14958	Human hmgI mRNA for high mobility group protein Y
29	T47342	PHOSPHOGLYCERATE MUTASE, BRAIN FORM (Homo sapiens)
30	X02492	INTERFERON INDUCED PROTEIN 6 16 PRECURSOR (HUMAN),contains L1 repetitive element ,
31	H11084	VASCULAR ENDOTHELIAL GROWTH FACTOR (Cavia porcellus)
32	U07158	Human syntaxin mRNA, complete cds
33	M76378	Human cysteine rich protein (CRP) gene, exons 5 and 6
34	U09564	Human serine kinase mRNA, complete cds
35	H06524	GELSOLIN PRECURSOR, PLASMA (HUMAN),
36	T47424	INSULIN RECEPTOR SUBSTRATE-1 (Homo sapiens)
37	H40095	MACROPHAGE MIGRATION INHIBITORY FACTOR (HUMAN),
38	Z49269	H sapiens gene for chemokine HCC 1
39	M19311	Human calmodulin mRNA complete cds
40	H61410	PLATELET GLYCOPROTEIN IV (Homo sapiens)
41	M92287	Homo sapiens cyclin D3 (CCND3) mRNA, complete cds
42	R08829	PYRUVATE KINASE, ISOZYMES R/L (Homo sapiens)
43	R73660	GAMMA-INTERFERON-INDUCIBLE PROTEIN IP 30 PRECURSOR (HUMAN),
44	X75208	H sapiens HEK2 mRNA for protein tyrosine kinase receptor
45	M91463	Human glucose transporter (GLUT4) gene, complete cds
46	X51416	Human mRNA for steroid hormone receptor hERR1
47	L11706	Human hormone sensitive lipase (LIPE) gene, complete cds
48	R84411	SMALL NUCLEAR RIBONUCLEOPROTEIN ASSOCIATED PROTEINS B AND B' (HUMAN),
49	T92451	TROPOMYOSIN, FIBROBLAST AND EPITHELIAL MUSCLE-TYPE (HUMAN),
50	L08069	Human heat shock protein, E coli DnaJ homologue mRNA, complete cds

Table 34. Top 50 Best genes with SVM-Soft for Golub Dataset using DF-LSGP approach

Rank	Gene ID	Gene annotation
1	M23197_at	CD33 CD33 antigen (differentiation antigen)
2	M30703_s_at	Amphiregulin (AR) gene
3	M31166_at	PTX3 Pentaxin-related gene, rapidly induced by IL-1 beta
4	M55150_at	FAH Fumarylacetoacetate
5	M31523_at	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
6	AFFX-HUMTFRR/M11507_M_at	AFFX-HUMTFRR/M11507_M_at (endogenous control)
7	L12168_at	ADENYLYL CYCLASE-ASSOCIATED PROTEIN 1
8	U82759_at	GB DEF = Homeodomain protein HoxA9 mRNA
9	M15205_at	TK1 Thymidine kinase 1, soluble
10	M14016_at	UROD Uroporphyrinogen decarboxylase
11	X85116_rna1_s_at	Epb72 gene exon 1
12	X74874_rna1_s_at	RNA polymerase II largest subunit gene extracted from H sapiens gene for RNA pol II largest subunit,
13	S76638_at	NFKB2 Nuclear factor of kappa light polypeptide gene enhancer in B-cells 2 (p49/p100)
14	M60974_s_at	DDIT1 DNA-damage-inducible transcript 1
15	U05259_rna1_at	MB-1 gene
16	D84145_at	WS-3 mRNA
17	L13852_at	UBE1L Ubiquitin-activating enzyme E1, like
18	HG2724-HT2820_at	Oncogene Tls/Chop, Fusion Activated
19	D88270_at	GB DEF = (lambda) DNA for immunoglobin light chain
20	X59350_at	CD22 CD22 antigen
21	U46751_at	Phosphotyrosine independent ligand p62 for the Lck SH2 domain mRNA
22	AFFX-HUMTFRR/M11507_3_at	AFFX-HUMTFRR/M11507_3_at (endogenous control)
23	M31303_rna1_at	Oncoprotein 18 (Op18) gene
24	U34877_at	Biliverdin-IXalpha reductase mRNA
25	M29610_s_at	GYPE Glycophorin E
26	Z14982_rna1_at	MHC-encoded proteasome subunit gene LAMP7-E1 gene (proteasome subunit LMP7) extracted from
27	X62535_at	DAGK1 Diacylglycerol kinase, alpha (80kD)
28	D13640_at	HLA C Major histocompatibility complex, class I, C
29	D26308_at	NADPH-flavin reductase
30	X95735_at	Zyxin
31	HG1612-HT1612_at	Macmarcks
32	D42043_at	KIAA0084 gene, partial cds
33	S67325_at	PCCB Propionyl Coenzyme A carboxylase, beta polypeptide
34	X57351_at	RPS3 Ribosomal protein S3
35	U77604_at	Microsomal glutathione S-transferase (GST-II) mRNA
36	M92287_at	CCND3 Cyclin D3
37	L07633_at	INTERFERON GAMMA UP-REGULATED I-5111 PROTEIN PRECURSOR
38	M27891_at	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
39	U23852_s_at	GB DEF = T-lymphocyte specific protein tyrosine kinase p56lck (lck) abberant mRNA
40	U62136_at	Putative enterocyte differentiation promoting factor mRNA, partial cds
41	M19283_at	ACTG1 Actin, gamma 1
42	X02596_at	GB DEF = Bcr (breakpoint cluster region) gene in Philadelphia chromosome
43	X51521_at	VIL2 Villin 2 (ezrin)
44	Y00764_at	ARH9 Aplysia ras-related homolog 9
45	U31248_at	ZNF174 Zinc finger protein 174
46	D63476_at	KIAA0142 gene
47	Z68228_s_at	JUP Junction plakoglobin
48	J03801_f_at	LYZ Lysozyme
49	M19045_f_at	LYZ Lysozyme
50	Z15115_at	TOP2B Topoisomerase (DNA) II beta (180kD)

Table 35. Top 50 Best genes for Alon with SVM-soft Dataset using DF-LSGP approach

Rank	Gene ID	Gene annotation
1	M76378	Human cysteine rich protein (CRP) gene, exons 5 and 6
2	R83354	GDP DISSOCIATION INHIBITOR FOR RHO PROTEIN (Bos taurus)
3	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6
4	H22579	INTEGRIN ALPHA-6 PRECURSOR (Homo sapiens)
5	Z50753	H sapiens mRNA for GCAP II/uroguanylin precursor
6	T94993	FIBROBLAST GROWTH FACTOR RECEPTOR 2 PRECURSOR (Homo sapiens)
7	R87126	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
8	X66975	H sapiens mRNA for heterogeneous nuclear ribonucleoprotein
9	H29546	NEUROTENSIN RECEPTOR (Homo sapiens)
10	R70939	TRANSCRIPTION FACTOR TAU 131 KD SUBUNIT (Saccharomyces cerevisiae)
11	T57882	MYOSIN HEAVY CHAIN, NONMUSCLE TYPE A (Homo sapiens)
12	J02854	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN),contains element TAR1
13	R08183	Q04984 10 KD HEAT SHOCK PROTEIN, MITOCHONDRIAL ,
14	U14631	Human 11 beta-hydroxysteroid dehydrogenase type II mRNA, complete cds
15	X14958	Human hmgI mRNA for high mobility group protein Y
16	R90908	PUTATIVE SERINE/THREONINE-PROTEIN KINASE T17E9 1 IN CHROMOSOME III (Caenorhabditis
17	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds
18	R48303	TYROSINE RICH ACIDIC MATRIX PROTEIN (Bos taurus)
19	M93010	Human epithelial cell marker protein 1 (HMe1) mRNA, complete cds
20	D15049	Human mRNA for protein tyrosine phosphatase
21	R36977	P03001 TRANSCRIPTION FACTOR IIIA ,
22	L13385	Homo sapiens(clone 71) Miller-Dieker lissencephaly protein (LIS1) mRNA, complete cds
23	T51023	HEAT SHOCK PROTEIN HSP 90-BETA (HUMAN)
24	L40904	H sapiens peroxisome proliferator activated receptor gamma, complete cds
25	R78927	COATOMER BETA' SUBUNIT (HUMAN),
26	R74208	GENERAL NEGATIVE REGULATOR OF TRANSCRIPTION SUBUNIT 4 (Saccharomyces cerevisiae)
27	H49870	MAD PROTEIN (Homo sapiens)
28	R44301	MINERALOCORTICOID RECEPTOR (Homo sapiens)
29	X70297	NEURONAL ACETYLCHOLINE RECEPTOR PROTEIN, ALPHA-7 CHAIN (HUMAN),
30	X12548	Human mRNA for lysosomal acid phosphatase (EC 3 1 3 2)
31	X80692	H sapiens ERK3 mRNA
32	M16827	Human medium-chain acyl-CoA dehydrogenase (ACADM) mRNA, complete cds
33	X16504	Human ENO3 mRNA for beta-enolase (EC 4 2 1 11)
34	R46739	HLA-B-ASSOCIATED TRANSCRIPT 3 (Homo sapiens)
35	X16354	Human mRNA for transmembrane carcinoembryonic antigen BGPa (formerly TM1-CEA)
36	L00352	Human low density lipoprotein receptor gene, exon 18
37	M22382	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN),
38	T51250	CYTOCHROME C OXIDASE POLYPEPTIDE VIII-LIVER/HEART (HUMAN)
39	D14662	Human mRNA for ORF, complete cds
40	R49542	OCS ELEMENT BINDING FACTOR 1 (Zea mays)
41	H64489	LEUKOCYTE ANTIGEN CD37 (Homo sapiens)
42	M87434	Human 71 kDa 2'5 oligoadenylate synthetase (p69 2-5A synthetase) mRNA, complete cds
43	R71401	HEMOGLOBIN ALPHA-1, ALPHA-2, AND ALPHA-3 CHAINS (Macaca assamenses)
44	T79831	MAP KINASE PHOSPHATASE 1 (Homo sapiens)
45	X89985	H sapiens mRNA for BCL7B protein
46	X76057	MANNOSE-6-PHOSPHATE ISOMERASE (HUMAN),
47	M63391	Human desmin gene, complete cds
48	L39874	Homo sapiens deoxycytidylate deaminase gene, complete cds
49	D30655	Human mRNA for eukaryotic initiation factor 4AII
50	H43887	COMPLEMENT FACTOR D PRECURSOR (Homo sapiens)

Table 36. Top 50 Best genes with SVM-Soft for Golub Dataset using Rec-TNoM approach

Rank	Gene ID	Gene annotation
1	M31523_at	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
2	M27783_s_at	ELA2 Elastatse 2, neutrophil
3	Z78285_f_at	GB DEF = mRNA (clone 1A7)
4	X95735_at	Zyxin
5	X94232_at	Novel T-cell activation protein
6	U46499_at	GLUTATHIONE S-TRANSFERASE, MICROSOMAL
7	U27460_at	Uridine diphosphoglucose pyrophosphorylase mRNA
8	M27891_at	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
9	X59417_at	PROTEASOME IOTA CHAIN
10	M23197_at	CD33 CD33 antigen (differentiation antigen)
11	M71243_f_at	GB DEF = Glycophorin Sta (type A) exons 3 and 4, partial
12	L09209_s_at	APLP2 Amyloid beta (A4) precursor-like protein 2
13	U26312_s_at	Heterochromatin protein HP1Hs-gamma mRNA
14	M84526_at	DF D component of complement (adipsin)
15	U10485_at	Lymphoid-restricted membrane protein (Jaw1) mRNA
16	HG1612-HT1612_at	Macmarcks
17	M33493_s_at	Tryptase-III mRNA, 3' end
18	M83652_s_at	PFC Properdin P factor, complement
19	X06182_s_at	KIT V-kit Hardy-Zuckerman 4 feline sarcoma viral oncogene homolog
20	X67491_f_at	GB DEF = Glutamate dehydrogenase
21	Z15115_at	TOP2B Topoisomerase (DNA) II beta (180kD)
22	M33680_at	26-kDa cell surface protein TAPA-1 mRNA
23	M92287_at	CCND3 Cyclin D3
24	X62654_rna1_at	ME491 gene extracted from H sapiens gene for Me491/CD63 antigen
25	M11722_at	Terminal transferase mRNA
26	U72936_s_at	X-LINKED HELICASE II
27	L49218_f_at	RB1 Retinoblastoma 1 (including osteosarcoma)
28	D88422_at	CYSTATIN A
29	M13690_s_at	C1NH Complement component 1 inhibitor (angioedema, hereditary)
30	M63138_at	CTSD Cathepsin D (lysosomal aspartyl protease)
31	X85116_rna1_s_at	Epb72 gene exon 1
32	M55150_at	FAH Fumarylacetoacetate
33	Z29067_at	Nek3 mRNA for protein kinase
34	X16699_at	CYP4B1 Cytochrome P450, subfamily IVB, polypeptide 1
35	M22960_at	PPGB Protective protein for beta-galactosidase (galactosialidosis)
36	U09087_s_at	Thymopoietin beta mRNA
37	U73738_at	GB DEF = Calcium/calmodulin-dependent protein kinase II delta E mRNA, partial cds
38	M32304_s_at	TIMP2 Tissue inhibitor of metalloproteinase 2
39	X76648_at	GLRX Glutaredoxin (thioltransferase)
40	U70063_at	Acid ceramidase mRNA
41	U25128_at	PTH2 parathyroid hormone receptor mRNA
42	X62320_at	GRN Granulin
43	X97335_at	Kinase A anchor protein
44	U20499_at	Estrogen sulfotransferase mRNA
45	U41635_at	OS-9 precurosor mRNA
46	X70297_at	CHRNA7 Cholinergic receptor, nicotinic, alpha polypeptide 7
47	D86967_at	KIAA0212 gene
48	Z17240_at	HMG2 High-mobility group (nonhistone chromosomal) protein 2
49	U05259_rna1_at	MB-1 gene
50	X03934_at	GB DEF = T-cell antigen receptor gene T3-delta

Table 37 Top 50 Best genes with SVM-Soft for Alon Dataset using Rec-TNoM approach

Rank	Gene ID	Gene annotation
1	M63391	Human desmin gene, complete cds
2	R87126	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
3	X53586	Human mRNA for integrin alpha 6
4	L12723	Human heat shock protein 70 (hsp70) mRNA, complete cds
5	T49647	MYRISTOYLATED ALANINE RICH C KINASE SUBSTRATE (Homo sapiens)
6	M76378	Human cysteine rich protein (CRP) gene, exons 5 and 6
7	J02854	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN),contains element TAR1
8	X12671	Human gene for heterogeneous nuclear ribonucleoprotein (hnRNP) core protein A1
9	X70297	NEURONAL ACETYLCHOLINE RECEPTOR PROTEIN, ALPHA 7 CHAIN (HUMAN),
10	R36977	P03001 TRANSCRIPTION FACTOR IIIA ,
11	M76378	Human cysteine rich protein (CRP) gene, exons 5 and 6
12	U26710	Human cbl b mRNA, complete cds
13	R77780	TRANSPOSABLE ELEMENT ACTIVATOR (Zea mays)
14	Z49269	H sapiens gene for chemokine HCC 1
15	M26697	Human nucleolar protein (B23) mRNA, complete cds
16	R42501	INOSINE-5 -MONOPHOSPHATE DEHYDROGENASE 2 (HUMAN),
17	X14958	Human hmgl mRNA for high mobility group protein Y
18	M80815	H sapiens a L fucosidase gene, exon 7 and 8, and complete cds
19	M90516	Human glutamine fructose 6 phosphate amidotransferase (GFAT) mRNA, complete cds
20	T92451	TROPOMYOSIN, FIBROBLAST AND EPITHELIAL MUSCLE-TYPE (HUMAN),
21	Z50753	H sapiens mRNA for GCAP-II/uroguanylin precursor
22	H22579	INTEGRIN ALPHA-6 PRECURSOR (Homo sapiens)
23	Z49269	H sapiens gene for chemokine HCC-1
24	L05144	PHOSPHOENOLPYRUVATE CARBOXYKINASE, CYTOSOLIC (HUMAN),contains Alu repetitive
25	M22382	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN),
26	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds
27	H85361	ATP BINDING CASSETTE TRANSPORTER 2 (Mus musculus)
28	U19796	Human melanoma antigen p15 mRNA, complete cds
29	L41559	Homo sapiens pterin 4a carbinolamine dehydratase (PCBD) mRNA, complete cds
30	R70790	GTP AMP PHOSPHOTRANSFERASE MITOCHONDRIAL (Rattus norvegicus)
31	M76378	Human cysteine rich protein (CRP) gene, exons 5 and 6
32	X63629	H sapiens mRNA for p cadherin
33	T67173	RETINOIC ACID RECEPTOR RXR BETA ISOFORM 2 (Homo sapiens)
34	K03474	Human Mullerian inhibiting substance gene, complete cds
35	M36634	Human vasoactive intestinal peptide (VIP) mRNA, complete cds
36	D14689	Human mRNA for ORF, complete cds
37	T51571	P24480 CALGIZZARIN
38	T90549	P SELECTIN PRECURSOR (Homo sapiens)
39	X74295	H sapiens mRNA for alpha 7B integrin
40	T86473	NUCLEOSIDE DIPHOSPHATE KINASE A (HUMAN),
41	R33367	MEMBRANE COFACTOR PROTEIN PRECURSOR (Homo sapiens)
42	H06524	GELSOLIN PRECURSOR, PLASMA (HUMAN),
43	H77597	H sapiens mRNA for metallothionein (HUMAN),
44	D14812	Human mRNA for ORF, complete cds
45	H40095	MACROPHAGE MIGRATION INHIBITORY FACTOR (HUMAN),
46	H43887	COMPLEMENT FACTOR D PRECURSOR (Homo sapiens)
47	X12369	TROPOMYOSIN ALPHA CHAIN, SMOOTH MUSCLE (HUMAN),
48	M91463	Human glucose transporter (GLUT4) gene, complete cds
49	L08069	Human heat shock protein, E coli DnaJ homologue mRNA, complete cds
50	X87342	H sapiens mRNA for human giant larvae homolog

REFERENCES

[1] Giora Unger and Benny Chor "Linear Separability of Gene Expression Datasets" IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol.7, No. 2, April-June2010.

[2] T.H. Bø and I. Jonassen, "New Feature Subset Selection Procedures for Classification of Expression Profiles," Genome Biology, vol. 3, no. 4,pp. 0017.1-0017.11, Mar. 2002.

[3] Corinna Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995.

[4] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeeck M,Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, *et al.*: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999, 286:531-537.

[5] Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, LevineAJ: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci USA* 1999, 96:6745-6750.

[6] Chris Ding, and Hanchuan Peng, "Minimum redundancy feature selection from microarray gene expression data," Journal of Bioinformatics and Computational Biology, Vol. 3, No. 2, pp.185-205, 2005.

[7] D.G. Beer et al., "Gene-Expression Profiles Predict Survival of Patients with Lung Adenocarcinoma," Nature Medicine, vol. 8, no. 8, pp. 816-824 Aug. 2002.

[8] A. Bhattacharjee et al., "Classification of Human Lung Carcinomas by mRNA Expression Profiling Reveals Distinct Adenocarcinoma Subclasses," Proc. Nat'l Academy of Sciences of the USA, vol. 98, no. 24, pp. 13790-795, Nov. 2001.

[9] Kohavi, R., & John, G. (1997).Wrapper for feature subset selection, *Artificial Intelligence, 97(1-2)*, 273-324.

[10] G.J. Gordon et al., "Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer and Mesothelioma," Cancer Research, vol. 62, no. 17, pp. 4963-4967, Sept. 2002.

[11] Amirali Jafarian and Alioune Ngom, "A New Gene Subset Selection Approach Based on Linear Separating Gene Pairs", IEEE International Conference on Computational Advances in Bio and medical Sciences (ICCABS 2011), Orlando FL, Feb 3-5, 2011, pp.105-110.

[12] Yvan Saeys, Inaki Inza and Pedro Larranaga,"A review of feature selection techniques in bioinformatics"Bioinformatics, Vol. 23 no 19 2007,pp 2507-2517

[13] Piyushkumar A.Mundra, JagathC.Rajapakse "Gene and sample selection for cancer classification with supportvectors based t-statistic"

[14] Isabelle Guyon, Jason Weston, Stephan Barnhill, M.D and Vladimir Vapnik "Gene Selection for Cancer Classification Using Support Vector Machines" Machine Learning 2002, 46(1-3):389-422.

[15] Amir Ben-Dor, Laurakay Bruhn, Nir Friedman, Iftach Nachman, Michel Schummer, Zohar Yakhini " Tissue Classification with Gene Expression Profiles" RECOMB 2000:54-64

[16] Duda, R. O. & Hart, P. E. (1973), *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York.

[17] Bin Chen and Guangri Quan. NP-hard Problems of Learning From Examples. In proceedings of 2008 IEE Fifth International Conference on Fuzzy and Knowledge Discovery.

[18] A Blum and P Langley. Selection of relevant features and examples in machine learnining. *Artificial Intelligence, 97:245-271, 1997*

[19] Y Saeys, I Lnza, and P Larranaga. A review of feature selection techniques in bioinformatics, *bioinformatics,* 23(19):2507-2517, 2007.

[20] H. Liu, H. Motoda, Instance Selection and Construction for Data Mining, Kluwer Academic Publishers, Boston, 2001.

[21] J. Doak, An evaluation of feature selection methods and their application to computer security, Tech. rep., Univ. of California at Davis, Dept. Computer Science (1992).

[22] Liu, H. Li, J. and Wong, L. A comparative study on feature selection and classification methods using gene expression profiles and proteomic pattern. Genomic Informatics, 13, 51-60, 2002

[23] Kira, K. and Rendell, L. A . The feature selection problem: Tradional methods and a new algorithm. In: Proceedings of Ninth National Conference on Artificial Intelligence, 129- 134, 1992

[24] Sinha,S. (2003) Discriminative motifs. J. Comput. Biol., 10, 599–615.

[25] P. Pudil, J. Novovitovi, and S. Bliha. Statistical Approach to Pattern Recognition: Theory and Practical Solution by Means **of** PREDITAS System. Kybernetika, 27:Supplement, 1-78, 1991.

[26] Hedenfalk I, Duggan D, Chen Y, Radmacher M, Bittner M, Simon R, Meltzer P, Gusterson B, Esteller M, Kallioniemi OP, Wilfond B, Borg A, Trent J, Raffeld M, Yakhini Z, Ben-Dor A, Dougherty E, Kononen J, Bubendorf L, Fehrle W, Pittaluga S, Gruvberger S, Loman N, Johannsson O, Olsson H, Sauter G.(2001). Gene-expression profiles in hereditary breast cancer. The New England Journal of Medicine, 344, 539-548. 2, 17

[27] Wang, Y. and Makedon, F. (2004). Application of Relief-f feature filtering algorithm to select informative genes for cancer classification using microarray data. In proceedings of the 2004 IEEE Computational System Bioinformatics, IEEE 13,24

[28] Liu, H. and Motoda, H. (1998). Feature extraction, construction and selection, A data mining prespective, Kluwer Academic Publishers. 10

[29] Félix F. Gonález Navarro and Lluís A. Belanche Muñoz. TFS: A Thermodynamical Search Algorithm for Feature Subset Selection. International Journal of Computer, Mathematical Science and applications. Special Issue on Advances in Computing and Optimization Techniques. Vol. 2 No. 1-2 PP. 123-135 2007.

[30] http://en.wikipedia.org/wiki/N-sphere

VITA AUCTORIS

Amirali Jafarian was born in Tehran, Iran in 1982; he obtained his BSc in Software Engineering from Azad University of Tehran South Branch. He is currently a candidate for a Master's degree in Computer Science at University of Windsor.