Noname manuscript No.
(will be inserted by the editor)

Improving the prediction of the clinical outcome of breast cancer using evolutionary algorithms

Mattias Wahde[1], **Zoltan Szallasi**[2]

[1] Department of Machine and Vehicle Systems, Chalmers University of Technology, 412 96 Göteborg, Sweden, e-mail: mattias.wahde@me.chalmers.se
[2] Children's Hospital Informatics Program, Harvard Medical School, Boston, MA, USA, e-mail: zszallasi@chip.org

Abstract There exist several methods for binary classification of gene expression data sets. However, in the majority of published methods, little effort has been made to minimize classifier complexity. In view of the small number of samples available in most gene expression data sets, there is a strong motivation for minimizing the number of free parameters that must be fitted to the data. In this paper, a method is introduced for evolving (using an evolutionary algorithm) simple classifiers involving a minimal subset of the available genes. The classifiers obtained by this method perform well, reaching 97% correct classification of clinical outcome on training samples from the breast cancer data set published by van't Veer, and up to 89% correct classification on validation samples from the same data set, easily outperforming previously published results.

Key words data classification – breast cancer – evolutionary algorithms

1 Introduction

Recent publications indicate that microarray-based gene expression signatures may in fact improve prediction of the clinical outcome of breast cancer, such as metastasis free survival [12], [13]. The predictor in these publications relies on the most relevant set of genes as determined by the correlation coefficient between the expression level of these genes and the disease outcome. Starting with the most relevant gene, more and more genes are added until no improvement is detected in predicting the disease outcome. Despite its initial success, reaching 83% accuracy [12], this method has an important limitation: Let us assume that there are two independent mechanisms responsible for the tumor phenotype causing early metastasis. If either of these mechanisms is activated, as reflected in, for example, the up-regulation of one of the genes involved and down-regulation of the other, short metastasis-free survival is detected. Let us further assume, that the first mechanism is activated in 70% of the short metastasis-free survival cases and the second mechanism is activated in 30% of the short metastasis-free survival cases. By simply considering the relevance of individual genes the second mechanism would not qualify as a potential classifier, although the two mechanisms together would predict all short metastasis-free survival cases.

Furthermore, in [12] and [13], as indeed in other publications on predicting the metastatic ability of solid tumors [11], the number of features (genes) involved in the classifiers is on the order of magnitude of about 10^2. In principle, it is possible that accurate prediction may require a predictor involving this many genes. However, this will greatly increase the cost of independent validation. Microarray measurements are known to produce strong platform specific biases, as reflected in the poor correlation of gene expression levels measured in the same RNA sample by two or more different microarray platforms [15]. Diagnostic decisions, however, should be based on gene classifiers that are not microarray platform specific and can be confirmed by independent methods such as quantitative RT-PCR. This serves as a strong incentive to reduce the number of classifying genes to the smallest reliable set. It should also be noted that, with limited training data available, a classifier involving few parameters is more likely to provide accurate predictions on unseen data than one involving many parameters.

The search for relevant combinations of complementary gene classifiers, however, suffers from combinatorial explosion. For example, an exhaustive search of combinations of 4 genes from a set of 10,000 measured genes involves testing 10^{16} subsets, a number beyond the reach of currently available computers. However, there exists a number of computational methods applicable to the search for a complementary set of predictor genes, e.g. support vector machines [2], evolutionary algorithms, etc. Evolutionary algorithms are particularly well suited

for searching through large search spaces. The use of such algorithms in classification based on gene expression data is relatively new, and has so far been limited to selecting the genes to be used in a classifier, rather than determining the classifier itself [8], [10], [3].

In this paper, a classification method that uses evolutionary algorithms[1] to search for linear classifiers will be introduced and applied to a breast cancer derived microarray based data set [12].

2 Method

Consider the problem of binary classification, i.e. assigning correct classes to the M samples of a gene expression data set, in which M_I elements belong to class I and M_{II} to class II.

2.1 Classifier definition

The main underlying assumption the method presented here is that there exists some function h such that the inequality

$$h(g_{i_1}, g_{i_2}, \ldots, g_{i_n}) > 0, \tag{1}$$

is satisfied for samples belonging to a class I, and not satisfied for samples in class II. The indices $i_1, \ldots, i_n$ determine which n genes are included in the classifier.

In principle, the function h may take any arbitrarily complex form, and i_j can take any value in the range $[1, N]$, where N is the number of genes measured in the data set. However, invoking Ockham's razor, motivated by the small size of the available data sets, a series expansion is made for h, resulting in

$$h(g_{i_1}, g_{i_2}, \ldots, g_{i_n}) \approx \beta + \sum_{j=1}^{n} \alpha_j g_{i_j} + O(g_{i_j}^2). \tag{2}$$

Thus, dropping higher-order terms, thereby reducing the number of unknown parameters, a linear classifier of the form

$$\beta + \sum_{j=1}^{n} \alpha_j g_{i_j} > 0 \tag{3}$$

is obtained. This type of classifier will be called a *linear, single-threshold classifier*. Thus, for any value of n, the number of unknown parameters equals $2n + 1$, of which n are the integer index values of the genes included in the classifier.

[1] Evolutionary algorithms (EAs) is an umbrella term incorporating various different algorithms, e.g. genetic algorithms (GAs), genetic programming (GP) etc. The algorithm used in this paper differs somewhat from the standard GA, and the abbreviation EA will therefore be used when referring to this algorithm.

2.2 Data preprocessing: relevance list

Clearly, selecting a few genes among thousands is a difficult task. However, not all genes are equally likely to be useful in classifiers. For example, a gene for which the expression values vary more or less randomly in samples of both classes is less likely to be useful. Thus, in the method presented here, a data preprocessing step is used in which the genes are ordered in a *relevance list* based on their performance as single-gene classifiers, i.e. classifiers of the form

$$\beta + g_{i_1} > 0. \tag{4}$$

Thus, at the top of the relevance list is the gene (or genes) that, by itself, best classifies the data set. It should be noted that useful genes *may* be found far from the top of the relevance list. Thus, the optimization method, which will now be described, selects genes from among the top L genes of the relevance list only with a certain probability, and otherwise selects genes completely randomly.

The ranking method employed when forming the relevance list is very similar to the Threshold Number of Misclassification (TNoM) score introduced in [1].

2.3 Optimization method

In the development of methods for data classification, one should distinguish between the *architecture* used for the classifiers (e.g. linear, single-threshold classifiers) and the *algorithm* used for finding those classifiers in large gene expression data matrices. Thus, with this nomenclature, a method is the union of an architecture and an algorithm. In the method introduced here, the algorithm for finding linear, single-threshold classifiers is an evolutionary algorithm (EA). EAs are methods for search and optimization inspired by natural evolution, and they are known to be particularly useful in large, complex search spaces with many local optima. For an introduction to EAs, see e.g. [9] or [4]. A comprehensive introduction to the use of EAs in bioinformatics can be found in [5].

A computer program (`EA_classifier`) for searching for linear, single-threshold classifiers as introduced in Eq. (3) has been written. In order to make clearer the distinction between the artificial genes used in the EA and the genes measured in the expression matrix, the former will henceforth be denoted *EA-genes*.

The search for classifiers is performed using a EA with variable chromosome length. The chromosomes encode, in their $2n + 1$ EA-genes, the identities of the n genes, the coefficients $\alpha_j, j = 1, \ldots, n$, and the coefficient β. Each EA-gene is a real-valued number in the range $[0, 1[$. During the decoding procedure, when an individual (a classifier) is formed from a chromosome, the value of each EA-gene is rescaled to an appropriate range (e.g.

$[-1, 1]$). The ranges of the coefficients α_j and β are set by the user.

The user also specifies the probability $p_{\rm r}$ to select, during mutation, genes from the relevance list (truncated to length L, also set by the user, typically in the range $[30, 100]$), rather than from the entire set of genes available in the gene expression matrix. Thus, if $p_{\rm r} = 1$, only genes present among the first L elements in the relevance list are allowed in the classifiers. Typically, $p_{\rm r}$ is chosen in the range $[0.6 - 0.8]$.

Each run of the EA begins with the construction of a stochastically generated population containing $N_{\rm p}$ chromosomes, each generated by selecting genes in the manner just described, and selecting parameters randomly in the allowed range. The number of EA-genes in each chromosome is also set to a random value ($2n+1$, where $n \in [n_{\rm min}, n_{\rm max}]$, with the two parameters $n_{\rm min}$ and $n_{\rm max}$ specified by the user).

The classifiers are obtained by decoding the $2n + 1$ EA-genes in the chromosome. Then, each classifier is evaluated and assigned a fitness score based on its ability to classify correctly the samples in the data set (the choice of fitness measure will be discussed below). When all classifiers have been evaluated, a new set of chromosomes, constituting the next generation, is formed through the processes of selection, crossover and mutation.

Selection is performed using the tournament method, in which two individuals are pitted against each other, and the best one (i.e. the one with the highest fitness score) is selected with probability $p_{\rm t}$. With probability $1 - p_{\rm t}$, the individual with lower fitness is chosen. Typically, $p_{\rm t}$ is chosen in the interval $[0.7, 0.8]$.

Crossover between two selected individuals is performed with probability $p_{\rm c}$, and *only* if the two individuals contain the same number of EA-genes. Thus, by only allowing crossover between individuals of equal size, the equivalent of species is introduced.

Mutations are performed in two fundamentally different ways: *structural mutations* change the size of the chromosome, whereas *parametric mutations* change the values of the EA-genes in the chromosomes.

Structural mutations, typically occurring with a low probability, either add (with probability 0.5) a gene, or remove a gene. Addition of a gene is achieved by adding two EA-genes: one for the identity of the gene, and one for the corresponding α-parameter. The EA-gene representing the parameter is chosen randomly in the allowed range, and the EA-gene representing the gene identity is chosen as described above, using the relevance list with probability $p_{\rm r}$. Parametric mutations for EA-genes representing gene identities are also performed in this way. For EA-genes representing α-parameters or the β parameter two different mutation methods are used: *full-range mutations*, in which the new value is selected randomly in the allowed range, and *creep mutations*, for which the new value is chosen in a narrow interval around the old value of the EA-gene in question.The various mutation probabilities are also set by the user.

Gene ID	# correct
4730	61
50	61
62	60
1	60

Table 1 The top four entries of the relevance list, based on the 78 samples in the training data set. The second column shows the number of correctly classified samples.

In addition to the procedures just outlined, *elitism* is used, i.e. each new generation includes one unchanged copy of the best individual from the previous generation.

While the user must to set quite a number of parameters before each run, the search procedure is not very sensitive to the exact values of most of the generic EA-parameters such as the population size, the crossover probability, the mutation rates (parametric and structural) etc., and the method is therefore very easy to use.

2.4 Fitness measures

As is well known, the results obtained from an EA are often strongly dependent on the choice of fitness measure. Thus, it is crucial that this choice be made in the best possible way.

For the classification problem studied here, the fitness measure can be defined in several ways. The simplest is to take the fitness f as

$$f_1(M_c) = \frac{M_c}{M} \tag{5}$$

where M_c is the number of correctly classified samples out of the total M samples. With this fitness measure, the EA will search for high-quality classifiers, but will not make any attempt to achieve maximal separation between the two classes. In fact, with the fitness measure f_1, the EA often produces classifiers such that several samples are located very near the separating hyperplane. An alternative fitness measure is given by

$$f_2(M_c, \overline{d}_c) = \frac{M_c}{M} + \epsilon_1 \overline{d_c}, \tag{6}$$

where $\overline{d_c}$ is the average distance between correctly classified samples and the separating hyperplane, and ϵ_1 is a small constant. The exact value of ϵ_1 is irrelevant, as long as it is small enough so that a classifier with $m + 1$ correctly classified samples always receives higher fitness than one with m correctly classified samples. A still more sophisticated fitness measure is

$$f_3(M_c, M_{c,r}, \overline{d_{c,r}}) = \frac{M_c}{M} - \epsilon_2 \left(M_{c,r} - \epsilon_3 \overline{d_{c,r}} \right), \tag{7}$$

where $\overline{d_{c,r}}$ is the average distance from the separating hyperplane of those correctly classified samples that are located within a range r of the same hyperplane, and $M_{c,r}$ is the number of such samples. ϵ_2 and ϵ_3 are two arbitrary, small positive constants. It is clear that this fitness measure will, unlike f_1 and f_2, specifically attempt to minimize the *number* of samples located within the range r of the separating hyperplane, and maximize the distance from that plane of those samples that it fails to remove from the range.

2.5 Data and Classifier Evaluation

The foremost test of the usefulness of a classifier involves testing its predictive capacity, i.e. its ability to classify correctly previously unseen samples.

In this paper, the simple procedure of dividing the data set into two subsets, a training set and a validation set, was employed. There is no strict solution to the problem of selecting the relative size of the two subsets. However, a common rule of thumb [6] is to use around 80% of the data for training and 20% for validation.

In this study, the data set published by van't Veer *et al.* [12], with 5,277 genes and 97 samples has been used. This data set was produced to extract classifiers from large-scale gene expression profiling of primary breast tumors with short and long metastasis free survival. A total of 51 samples in this data set belong to the class of long (over 5 years) metastatis free survival (arbitrarily denoted class I), whereas the remaining 46 samples belong to the class of short metastasis free survival (class II).

Using the above rule of thumb, a training set with 78 samples (of which 44 in class I) and a validation set with 19 samples (7 in class I), were obtained.[2] The expression values for all genes were normalized to the range $[-2.0, +2.0]$ using the minimum and maximum expressions over all data samples. Note that, in order to generate the data set with 5,277 genes, the raw data was filtered according to the instructions provided by van't Veer *et al.* [12]. All gene indices used henceforth in the paper refer to the filtered data. The mapping from gene indices to actual gene names can be found at www.me.chalmers.se/~mwahde/EAclassifier.htm.

3 Results

The `EA_classifier` computer program searches for classifiers according to the method outline above. The simplest possible classifier would be of the form $g_i > \beta$, where the index i and the threshold β are initially unknown parameters. The indices i of the best single-gene classifiers, as well as the number of correctly classified samples, are shown in Table 1. Such a list, which can be obtained in a matter of seconds from the computer program, constitutes the *relevance list* introduced above, which is used for selecting genes in the multi-gene classifiers. As can be seen in the table, genes 4730 and 50 top the list, with 61 correctly classified samples (78.2%).

[2] In general, the superscript T is used to refer to the training set, whereas the superscript V refers to the validation set.

n	M_c^{T}	% correct	n	M_c^{T}	% correct
2	68	87.2%	6	74	94.9%
3	71	91.0%	7	76	97.4%
4	74	94.9%	$\geq$8	$\leq$76	97.4%
5	74	94.9%			

Table 2 Classifier performance as a function of n: The second column shows the number of correctly classified samples in the training set (M_c^{T}) for the best classifiers found, as a function of n. For some $n > 2$, several different such classifiers were found.

3.1 Two-gene classifiers

In principle, the EA can be applied to the search for two-gene classifiers. However, for such classifiers, it is actually possible to perform an exhaustive search for the best gene pair. In this case, the search procedure runs through all $N_g(N_g-1)/2$ gene pairs, and determines the best parameters α_1, α_2, and β for each gene pair.

In order to find the best parameter values for any given gene pair, a projection technique is used, in which the two-gene distribution is projected onto a line inclined at an angle γ relative to the horizontal axis the first gene in the pair, generating a one-dimensional projected distribution, the performance of which can easily be determined. As the line rotates through the angular interval $\gamma \in [0, 2\pi[$ all possible projections are tested, and the best parameter set for the gene pair can thus be determined.

For the data set used here, with 5,277 genes and thus 13,920,726 distinct gene pairs, the exhaustive search for $n = 2$ lasted approximately 48 CPU hours on a 2.53GHz P4 computer.

The best 2-gene classifier thus found, achieving 68 correctly classified samples (87.2%), is shown in the top row of Table 3.

3.2 Multi-gene classifiers

A number of runs were performed with the aim of finding the best $n-$gene classifiers, for various (small) values of n. Table 2 shows the number of samples classified correctly, for different values of n, by the best classifiers found. In Table 3, detailed information about some of those classifiers is shown.

Name	n	Classifier	Result
C_1^2	2	$-0.5090g_1 + 0.8607g_{689} > 0.0779$	87.2%
C_1^3	3	$-0.5248g_{13} - 0.6366g_{1278} + 0.5651g_{3353} > -0.0129$	91.0%
C_2^3	3	$-0.4355g_1 + 0.8270g_{689} - 0.3555g_{4148} > 0.0894$	91.0%
C_3^3	3	$-0.4283g_1 + 0.6286g_{61} - 0.6492g_{5247} > 0.1275$	91.0%
C_1^4	4	$-0.3153g_1 - 0.3381g_2 + 0.6517g_{689} - 0.6013g_{4723} > 0.0873$	94.9%
C_1^5	5	$-0.2960g_{13} - 0.4546g_{42} - 0.4900g_{46} + 0.4584g_{62} - 0.5054g_{4401} > 0.1189$	94.9%
C_1^6	6	$-0.4688g_1 + 0.5259g_{50} + 0.4569g_{62} - 0.4816g_{391} - 0.2346g_{819} - 0.0890g_{5247} > 0.2693$	94.9%
C_2^6	6	$-0.3023g_1 + 0.5954g_{50} + 0.5171g_{62} - 0.2137g_{391} - 0.2811g_{879} - 0.4026g_{1917} > 0.2693$	94.9%
C_3^6	6	$-0.3215g_1 + 0.5133g_{61} + 0.6458g_{689} - 0.3627g_{4148} - 0.0823g_{4574} - 0.2788g_{5247} > 0.1309$	94.9%
C_4^6	6	$-0.5243g_1 - 0.4239g_8 + 0.5188g_{689} + 0.3226g_{1305} + 0.0191g_{2509} + 0.4145g_{5120} > 0.0619$	94.9%
C_1^7	7	$-0.3504g_1 - 0.1534g_{13} - 0.1107g_{47} + 0.5495g_{61} + 0.5200g_{689} + 0.0959g_{1989} - 0.5097g_{5247} > 0.1007$	97.4%

Table 3 Structure of the best classifiers for various values of n. In all cases, the classifiers have been normalized so that the sum of the squares of the coefficients is equal to 1.

Name	M_c^{T}	$M_{c,0.02}^{\mathrm{T}}$	$\overline{d}_c^T$	M_c^{V}	$M_{c,0.02}^{\mathrm{V}}$	$\overline{d}_c^V$
C_1^3	71	6	0.1629	13	4	0.1563
C_2^3	71	6	0.2208	13	1	0.2018
C_3^3	71	4	0.2724	15	1	0.2884
C_1^4	74	7	0.1986	14	3	0.1507
C_1^5	74	6	0.2430	13	2	0.2641
C_1^6	74	5	0.3264	12	2	0.2340
C_2^6	74	3	0.2527	13	1	0.2091
C_3^6	74	4	0.2361	16	2	0.2011
C_4^6	74	4	0.2931	13	3	0.2686
C_1^7	76	5	0.2611	17	0	0.2249

Table 4 Validation performance: The second and fifth columns of each row show the performance of the corresponding classifier on the training and validation sets, respectively. The third and sixth columns show the number of samples located within a distance 0.02 of the separating hyperplane, in the training and validation sets, respectively. The fourth and seventh columns show the root mean square distance between the separating hyperplane and correctly classified samples in the training and validation set, respectively.

For several $n > 2$, many classifiers were found that achieved the maximum values (for each n) shown in Tables 2 and 3. However, classifiers not shown in Table 3 involved the same combinations of genes as the ones shown, but with slightly different parameters. Henceforth, classifiers will be called *distinct* if they involve different combinations of genes and, with one exception (see Table 5), only distinct classifiers are shown in the tables.

Furthermore, while the search was exhaustive for $n = 2$, it becomes ever less so as n increases. For example, for $n = 10$, the selection of the 10 genes among the total of 5,277 can be made in 4.6×10^{30} different ways of which, naturally, only very few are evaluated in any given run. The ease by which the maximum number of correctly classified training samples is reached varies with n. For example, for $n = 4$, only a single distinct classifier with maximum performance was found, whereas for $n = 3$, three such classifiers were found. However, these classifiers varied in their performance on the validation set: C_1^3, C_2^3, and C_3^3, classified 13, 13, and 16 validation samples correctly, respectively.

Several long runs were made using many (more than 7) genes. However, in no case was a classifier found that performed better (during validation) than C_1^7.

In the runs reported so far, the fitness measure f_1 was used. Other than measuring the number of correctly classified samples, f_1 makes no attempt to judge the quality of the classifier, and therefore the variability in the performance on the validation set is not surprising. In order to judge the quality of a classifier, it is important to study the distance between the samples and the separating hyperplane. The results of such an investigation are shown in Table 4. For the purposes of evaluating the quality of classifiers, the following definition is introduced: the region within a given distance r of the separating hyperplane is called the *gray zone*, and samples located in this zone are called *gray zone samples*.

In the third column of the table, the number of samples *in the training set* located in the gray zone (the width of which was chosen somewhat arbitrarily to be 0.02) is shown. For classifiers derived using the f_1 fitness measure, there appears to be no advantage for a classifier to achieve a large root mean square distance between the samples and the separating hyperplane, except for the case $n = 3$.

3.3 Classifiers with maximum separation

While the results shown in Table 4 do not indicate a strong influence on classifier performance of either the average distance $\overline{d}_c^T$ or the number of samples in the gray zone ($M_{c,0.02}^{\mathrm{T}}$), the issue should be investigated more thoroughly (as the f_1 fitness measure completely ignores these two possible criteria). Thus, additional runs were made, using the fitness measures f_2 and f_3. Some results from these runs are shown in Tables 5 and 6. In general, the selection of genes for these runs was based on the results shown in Table 3. As can be seen from Table 6, fitness measures f_2 and f_3 resulted in a slight improvement in the average separation of the samples from the separating hyperplane. The validation performance was also improved in several cases. For example, classifier $\tilde{C}_{1,I}^4$ achieved 16 correct classifications in the validation set, compared to 14 for C_1^4. Furthermore, using f_3, the total number of samples (in the training and validation sets) in the gray zone decreased (e.g. from 10 to 7, going from C_1^4 to $\tilde{C}_{1,II}^4$). The most dramatic effect occured in the 7-gene classifier $\tilde{C}_{1,I}^7$ for which *no* samples remained in the gray zone of width 0.02. The gray zone width was increased to 0.05 in an attempt to improve further the classifier performance. The EA managed to press down the number of training set samples in this wider gray zone to 4. However, 3 of these were located within a distance 0.02 from the separating hyperplane. Thus, no further improvement was obtained.

3.4 Targeted runs

A few additional runs were made, in which a customized relevance list was used, consisting of all the genes appearing in the best classifiers in Table 3. The relevance list usage parameter was set to $p_r = 0.80$ in some runs, and $p_r = 1.00$ in other runs. n was allowed to take any value in the range 3 to 26 (the number of distinct genes in Table 3). In one such run, a 26-gene classifier achieving 77 correctly classified samples (98.7%) was in fact obtained. However, the performance on the validation set was abysmal: only 10 correctly classified samples (52.6%), indicating a strong case of overfitting.

4 Analysis and Discussion

4.1 Classification using EAs: related work

While evolutionary algorithms are rapidly becoming increasingly widespread in many fields of science, their use in classification of gene expression matrices has so far been very limited.

Ooi and Tan [10] considered the problem of multiclass classification, and used a GA to select genes that were then used in a maximum-likelihood classification method. Li *et al.* [8], have used a GA for selecting genes that are subsequently used in a k nearest neighbor (KNN) classifier. The number of genes selected by the EA was set to between 5 and 50. The GA was then run many (10^4) times, and genes were selected based on their frequency of appearance in the classifiers thus obtained. Their method also involves a post-processing step, in which the number of (ranked) genes to be used is selected. Deutsch [3] developed a GA-based method for minimizing the number of genes selected for inclusion in classifiers. Applying this method, Deutsch achieved an order-of-magnitude reduction in the number of genes needed for classification of small round blue cell tumors, arriving at a classifier containing of order 10^1 genes as compared to of order 10^2 genes in the classifier reported by Khan *et al.* [7]. However, as Li *et al.* [8], Deutsch used the GA only for finding the relevant genes, and then employed the KNN method for the classification. By contrast, in the present paper, not only the relevant genes but also the classification rule is obtained by the evolutionary algorithm.

4.2 Classifier complexity

The fact that the best classifier found by the method introduced here used only 7 genes justifies the basic idea of this paper, namely to look for classifiers with as few genes as possible. On the other hand, it also indicates the difficulty (due to combinatorial explosion) of searching the space of classifiers for large n. Thus, it cannot be excluded that a classifier with better performance than C_1^7 could be found. However, any improvement over C_1^7 would be slight, since this classifier comes close to perfect classification for both the training and validation sets. Note also that, already with 3-4 genes, excellent classifier performance can be achieved.

4.3 Classifier structure

It is interesting to note that the best classifiers to some extent have similar structure, with classifiers with larger n building upon their small-n counterparts. For example, all but two classifiers in Table 3 contain gene 1, with a negative coefficient, and 6 of the 11 classifiers in the same table contain gene 689, always with a positive coefficient. Gene 5247 appears in 4 of the classifiers, always with a negative coefficient, and gene 61 appears in 3 of the 11 classifiers, always with a positive coefficient.

Note that this state of affairs was not enforced by the computer program: each run was started from a random set of classifiers. The presence of gene 1 is perhaps not so surprising, given its location near the top of the relevance list (see Table 1). However, gene 5247, with 57 correctly classified samples (taken as a single-gene classifier), is located in the range $[27, 44]$ in the relevance list (together with 22 other genes that also achieve 57

Name	n	Classifier	Result
$\tilde{C}^3_{3,I}$	3	$-0.4524g_1 + 0.6263g_{61} - 0.6349g_{5247} > 0.1314$	91.0%
$\tilde{C}^4_{1,I}$	4	$-0.3739g_1 - 0.3868g_2 + 0.5662g_{689} - 0.6246g_{4723} > 0.0900$	94.9%
$\tilde{C}^4_{1,II}$	4	$-0.3186g_1 - 0.3767g_2 + 0.6113g_{689} - 0.6188g_{4723} > 0.0903$	94.9%
$\tilde{C}^7_{1,I}$	7	$-0.3465g_1 - 0.0885g_{13} - 0.2066g_{47} + 0.5417g_{61} + 0.5234g_{689} + 0.1305g_{1989} - 0.4950g_{5247} > 0.1233$	97.4%

Table 5 Structure of the best classifiers found for various values of n, using fitness measures f_2 and f_3. In all cases, the classifiers have been normalized so that the sum of the squares of the coefficients is equal to 1.

Name	M_c^{T}	$M_{c,0.02}^{\mathrm{T}}$	$\overline{d}_c^T$	M_c^{V}	$M_{c,0.02}^{\mathrm{V}}$	$\overline{d}_c^V$	Fitness	Gray zone range
$\tilde{C}^3_{3,I}$	71	3	0.2760	16	1	0.2937	f_2	–
$\tilde{C}^4_{1,I}$	74	7	0.2194	16	2	0.1474	f_2	–
$\tilde{C}^4_{1,II}$	74	3	0.2034	15	4	0.1457	f_3	0.02
$\tilde{C}^7_{1,I}$	76	0	0.2638	17	0	0.2330	f_3	0.02

Table 6 Validation performance, using the f_2 and f_3 fitness measures. The columns correspond to those in Table 4. The last two columns give information concerning the fitness measure used. Classifiers denoted $\tilde{C}^i_{j,k}$ contain the same genes as the corresponding classifier C^i_j in Table 3.

correctly classified samples), and genes 689 and 61, with 56 correctly classified samples, are located in the interval $[45, 106]$.

4.4 Selected genes

In the runs leading to the results in Table 3 (for $n > 2$), the truncation L of the relevance list was set either to 30 or 100, and genes were selected (during mutations in the EA) from this list with probability $p_r = 0.80$. One of the strenghts of the method introduced here is that it the selection of genes is *not* limited to the first L genes in the relevance list. In fact, in many of the top-performing classifiers, one or several genes from far down the relevance list were included (e.g. gene 3353 in C^3_1, which is number 1521 on the list).

4.5 Biological relevance

One of the main motivations to develop a reliable method for discovering few-gene classifiers has been the well-documented platform specific bias of microarray based gene expression measurements [15]. Due to this bias, a classifier derived using a given platform, such as the one used by van't Veer *et al.* [12], cannot be transferred onto data sets using a different microarray technology. Reliable classification, however, should preferably be based on longitudinal studies using widely and readily reproducible gene expression measurements such as quantitative RT PCR. Our method provides a more accurate classifier for metastasis-free survival with one order of magnitude fewer genes than the one provided by van't Veer *et al.* [12].

The method presented here provides only an efficient analytical framework to produce classifiers with improved accuracy and the actual results presented here should be placed into the context of the appropriate biological system analyzed. Binary classification involves a somewhat arbitrary discretization of the samples. In this case, short or long metastasis-free survival was defined as less or more than 5 years without detected metastasis. Considering the cases with 4.9 years and 5.1 years of metastasis-free survival falling into different categories the binary classifier will without doubt lead to a certain level of overfitting.

However, our method can readily be modified to accommodate a continuous classifier or possibly classification based on fuzzy logic. Work along these lines is already in progress. The suggested biological meaning of the classifiers should also be pointed out: The data set examined contains log ratio values - therefore negative and positive coefficients in a classifier would indicate decreased or increased expression of the corresponding genes. The criteria of increased and decreased gene expression levels in tumors with varying malignancy can be readily associated with the corresponding biology - for example, the well-known combinatorial involvement of increased levels of oncogenes and decreased levels of tumor suppressors in cancer development [14].

5 Conclusion

In this paper, a method for binary classification of gene expression data has been introduced and applied to the breast cancer data set obtained by van't Veer *et al.* [12]. The method makes use of a EA to search through the vast space of possible classifiers. The algorithm operates with a varying chromosome length, so that classifiers of different size can be handled.

The best classifier found contained 7 genes, but useful classifiers with fewer than 5 genes were discovered as well. It was also found that the validation performance of classifiers could be enhanced further by forcing the EA to

search for classifiers minimizing the number of samples in the immediate vicinity of the separating hyperplane.

Acknowledgments

This work was in part supported by the National Institutes of Health grants U01 HL66805-01 (NHLBI), U 01 HL066582-01 (NHLBI), and 1PO1CA 092644-01 (NCI) and by a grant from Vetenskapsrådet (The Swedish Research Council).

References

1. **Ben-Dor A et al** (2000) Tissue classification with gene expression profiles, J. Comp. Biol. **7**: 559-584
2. **Brown MP et al** (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines, Proc Natl Acad Sci U S A. **97**(1): 262-267
3. **Deutsch JM** (2003) Evolutionary algorithms for finding optimal gene sets in microarray prediction, Bioinformatics **19**: 45-52
4. **Fogel DB** (1999) Evolutionary computation: toward a new philosophy of machine intelligence, 2nd Ed. Wiley-IEEE Press
5. **Fogel G, Corne DW** (2002) Evolutionary computation in bioinformatics, Morgan Kaufmann.
6. **Haykin S** (1998) Neural Networks: A Comprehensive Foundation, 2nd Ed. Prentice-Hall
7. **Khan J et al** (2001) Classification and diagnostic prediction of cancers using expression profiling and artificial neural networks, Nat. Med. **7**: 673-679
8. **Li L et al** (2002) Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method, Bioinformatics **17**: 1131-1142
9. **Mitchell M** (1996) An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.
10. **Ooi CH, Tan, P** (2003) Genetic algorithms applied to multi-class prediction for the analysis of gene expression data, Bioinformatcs **19**: 37-44
11. **Ramaswamy S et al** (2003) A molecular signature of metastasis in primary solid tumors, Nature Genetics **33**(1): 49-54
12. **van't Veer LJ et al** (2002) Gene expression profiling predicts clinical outcome of breast cancer, Nature **415**: 530-536.
13. **van de Vijver MJ et al** (2002) A gene-expression signature as a predictor of survival in breast cancer, New England Journal of Medicine **347**: 1999-2009
14. **Vogelstein B, Kinzler KW** (1998) The genetic basis of human cancer. McGraw-Hill
15. **Yuen T et al** (2002) Accuracy and calibration of commercial oligonucleotide and custom cDNA microarrays, Nucleic Acids Res. **30**(10): e48