
Class Prediction Based on Gene Expression: Applying Neural Networks via a Genetic Algorithm Wrapper

Benjamin Good

Veterans Medical Research
Foundation
3350 La Jolla Village Drive
San Diego, CA 92161, USA
ben@schoolid.com

Jeremy Peay

Veterans Medical Research
Foundation
3350 La Jolla Village Drive
San Diego, CA 92161, USA
jpeay@vapop.ucsd.edu

Satish Pillai

Department of Biology
University of California
San Diego,
La Jolla, CA 92093-0679
satish@biomail.ucsd.edu

Jacques Corbeil

Department of Medicine
University of California
San Diego,
La Jolla, CA 92093-0679
and the Veterans Medical
Research Foundation
jcorbeil@ucsd.edu

Abstract

This project focuses on applying neural networks to the classification of biological state based on gene expression data. In order to take advantage of the non-linear classification abilities of neural networks, a genetic algorithm is employed as a “wrapper” feature selector. Results indicate that the genetic algorithm effectively identifies features that allow successful neural network training. In addition, it is shown that ensembles created by combining neural networks from multiple runs of the genetic algorithm consistently outperform single networks.

1 INTRODUCTION

Due to recent advances in biotechnology, gene expression can now be quantitatively monitored on a global scale. Currently, the activity of approximately 12,000 genes can be measured simultaneously. Within the near future, we will have the ability to monitor all of the estimated 35,000 genes of the human genome. The information present in this data will help in elucidating molecular mechanisms underlying phenotypes in both normal and disease states. This information will be useful for diagnosis and prognosis, and it may shed light on numerous biological processes related to pathogenesis. In order to take full advantage of this new source of information, methods for identifying those genes or groups of genes that are most pertinent to a given phenomenon must be identified. Standard statistical methods can identify simple correlations, but more advanced techniques are required to find complex

relationships between genes associated with different biological phenomena.

The goal of our project is to produce a classifier for gene expression studies that not only identifies genes whose expression is correlated with a class distinction, but also identifies potentially non-linear interactions between genes associated with class differences. This will hopefully produce a reliable classifier that works using only a small number of genes. Such a streamlined classifier will be particularly important for potentially high throughput applications such as diagnosis and prognosis because reducing the number of genes that need to be monitored significantly reduces the cost of the procedure. In addition, identifying a relatively small subset of genes that exhibit a consistent pattern of expression associated with a biological condition will serve as an excellent starting point in understanding the molecular mechanisms underlying that condition.

With non-linearity as a pre-condition, two well known types of classifiers are classification trees and multi-layer neural networks. Classifier trees are appealing because of the relative ease of explaining their behavior, but since they have already been investigated for use with gene expression data, without any improvement over linear methods, we chose to focus the present study on multi-layer neural networks (Dudoit *et al* 2000).

With thousands of potential features, extremely few available samples, and computationally intensive neural network training, the choice of feature selection method is a crucial and challenging task. We chose a genetic algorithm “wrapper” approach because it provides an efficient search mechanism directly tailored to the neural networks and capable of rewarding complex relationships present in the data (Yang and Hanovar

1997). The added number of cycles required by the wrapper approach as opposed to a “filtering” approach was judged necessary because most filtering methods do not take into account potentially useful feature interactions and may thus perform poorly in conjunction with a non-linear classifier. Since the primary reason for using neural networks in this application is the hope of discovering and exploiting complex interactions, it is essential to come up with a feature selection routine that does not eliminate them.

1.1 RELATED WORK

The idea of applying genetic algorithms as feature selectors is not novel (Chtioui, Bertrand, & Barba 1998),(Yang & Hanovar 1997), (Siedlecki & Sklansky 1989), and (Hallinan & Jackway 1999). Of these, Yang & Hanovar and Hallinan & Jackway investigate combinations of genetic algorithms and neural networks. Our project extends these and other author’s work by applying the general technique to a substantially different problem domain. The primary difference between gene expression data and all of the datasets used in the above publications is that our data has roughly 100 times more available features but much fewer sample cases. Thus the need to identify an efficient feature search methodology is obvious.

2 METHODS

2.1 DATA

For the experiments described in this paper, each sample consists of the expression levels of 6,817 genes as measured by an Affymetrix GeneChip¹ and the appropriate cancer classification. These classifications are based on current pathological techniques used in cancer diagnosis that utilize tumor morphology, cell surface markers, and cytogenetic analysis. These classification criteria are not absolute and misclassifications can occur. Furthermore, the criteria for classification are always being remolded as more is learned. Values used to represent gene expression range from 0 to 60,000. It is important to note that these values are not quantitative between two different genes, due to the kinetics of DNA hybridization, but are quantitative between the same gene in different samples (Naef *et al* 2001). When the words “inputs” or “features” are used, they refer to genes. The data from these features are the expression levels of those genes.

2.2 ALGORITHM

The version of our algorithm that is presented here employs a GA to search for useful combinations of inputs and network architectures². Neural networks trained with back propagation are used to perform the classification (and fitness evaluation).

The algorithm runs as follows:

- 1) Initialize a population in which each individual specifies a random subset of the available inputs, a random number of inputs to use, and a random number of hidden units.
- 2) Evaluate each individual in the population by producing a network with the specified inputs and number of hidden units, randomly initializing the network weights, training the network with the gene expression values of the specified inputs, and then running the trained network on an evaluation set consisting of the network training set and an additional unseen test set. The fitness for each individual is determined by the number of errors that the trained network makes on the evaluation set.
- 3) Select and reproduce.
- 4) Iterate 1-3 until a network is found that correctly classifies all of the samples in the evaluation set or a maximum number of generations is reached.
- 5) Test the best network on a third separate validation set.

2.3 ARTIFICIAL GENOME

The genotype employed includes an array of integers representing a set of inputs to the neural network, the number of inputs to use, and the number of hidden units of the neural network. No input may be used for more than one input to the neural network. This input diversity is enforced throughout evolution. This integer feature representation is different from the great majority of GA feature extraction projects in that other projects typically use a string with binary values representing whether or not a given feature is to be used by the classifier. This approach would of course be unfeasible in a domain with many thousands of potential features.

2.4 NEURAL NETWORKS

The networks are all fully connected feedforward with one hidden layer. The only aspects of network architecture that vary are the number of hidden units

¹ <http://www.affymetrix.com>

² The only aspects of architecture that change are the number of hidden units and the number of inputs.

and the number of inputs. A maximum number of hidden units and of inputs is set before the trial begins. As all of the classifications are made between only two classes at a time, all of the neural networks have only one output unit. All the nodes in the networks employ a sigmoid activation function that yields a floating point number in the range of 0 to 1. If the output of the output unit is greater than 0.5, the pattern is classified in one class, otherwise it is classified in the other.

The networks are trained using back propagation with momentum (Rumelhart *et al* 1986)(Plaut *et al* 1986). Unless otherwise noted, the networks were trained for 5 steps each evaluation, with a learning rate of 0.9, and a momentum of 0.8. The weights are updated after each pattern is presented, so there are 5 * number-of-training-cases weight adjustments for each individual evaluation.

These parameter choices were made primarily because they seemed to provide close to the minimum number of cycles needed to perform any meaningful learning. As the complexity of the patterns to be learned increases, we expect to have to increase the number of training steps and potentially decrease the learning rate³. These parameters might also be incorporated into the GA with an added fitness penalty for excessive processing time.

2.5 EVALUATION

One aspect of this project that seems to defy conventional wisdom is the low number of evaluations for each set of features. Since the performance of the network is determined by its inputs, architecture and initial random weights, many authors suggest evaluating with as many as 30 different random initializations in order to get a good indication of the input set's actual performance (Setiono & Liu 1997). But, as multiple training iterations add significantly to the time needed to run the algorithm, the number of evaluations was initially set to 1. Despite the noisy fitness evaluation, this method did produce positive results and increasing the number of evaluations per input set did not notably alter the number of generations needed to maximize the fitness function or alter the characteristics of the resultant networks.

Another difference between this project and previous work is the inclusion of an evaluation set separate from the network training set and in addition to the validation set. This additional division was included with hopes of providing the GA with pressure to select network/input sets that provided good generalization.

³ Other alternatives include network growing algorithms (Yang & Hanovar 1997) and the incorporation of weight training into the GA (Hallinan & Jackway 1999).

2.6 REPRODUCTION – TOURNAMENT SELECTION

After each individual has been assigned a fitness based on its post-training evaluation performance, the population is reproduced via tournament selection with replacement. Each individual in the population is paired with another randomly assigned member of the population and their fitnesses are compared. The individual with the lower fitness is judged the loser and is replaced by a mutated version of the winner. The winner stays in the population unchanged. Initial runs did not employ crossover.

2.6.1 Mutation

There are three separate mutation rates, one for number of inputs, one for number of hidden units, and one for the inputs themselves. Typical settings might be 0.25, 0.4, and 0.01 respectively. If a random mutation is indicated for the number of hidden units or for the number of inputs, the loser inherits the number from the winner, otherwise the loser maintains their original number⁴. The probability of mutation for each of the inputs is set by the input mutation rate. If an input mutation is indicated, that input is replaced by another randomly selected input.

If there are fewer inputs for the loser than the winner, some of the winner's inputs are not transferred. If there are more inputs for the loser than the winner, then some of the loser's inputs are retained.

For example:

Individual A inputs(23 900 6645 4 3000)
number of inputs(5) hidden(4)
Fitness(A) = 8

Individual B inputs(2 35 5)
number of inputs(3) hidden(7)
Fitness(B) = 34

Fitness(A) < Fitness(B) therefore B is the winner and A is to be replaced.

First, assign number of hidden units:

Rand(0,1) might return 0.7 which is greater than the hidden mutation rate of 0.4 so replace the number of hidden units of A with B.

hidden(A) = hidden(B) = 7

⁴ A better term for these two parameters might be "inheritance rate".

Now, assign number of inputs:

Rand(0,1) might return 0.5 which is greater than 0.1 so leave A with its original number of inputs.

Now, assign inputs

Inputs(B) = (2 35 5)

Inputs(A) = (23 900 6645 4 3000)

Inputs(A)* = (2 35 5 4 3000)

Now mutate inputs

Inputs(A) = (2 6942 5 4 3000)

So, after this round of the tournament:

(original)

A inputs(**23 900 6645** 4 3000)

number of inputs(5) hidden(**4**)

B inputs(2 35 5)

number of inputs(3) hidden(7)

(next gen)

A inputs(**2 6942 5** 4 3000)

number of inputs(5) hidden(7)

B inputs(2 35 5)

number of inputs(3) hidden(7)

This algorithm was chosen because it is very simple to implement and initial tests proved that it works reasonably well. As this project moves forward, the mechanics of the GA will continue to be refined.

2.7 DATASET

The dataset principally discussed in this paper is the one associated with Golub (1999) and published online at www.genome.wi.mit.edu/MPR. It consists of 72 samples consisting of the expression levels of 6817 genes. Of the 72 samples, 47 were from patients diagnosed with Acute Lymphoblastic Leukemia (ALL) and the remaining 25 were from patients diagnosed with Acute Myeloid Leukemia (AML). The ALL samples were again divided between 9 T-cell samples and 38 B-cell samples.

3 RESULTS

For all of the trials presented here, three sets had to be constructed. A training set for the neural networks, an evaluation set for the GA, and a validation set for the algorithm as a whole.

In runs without a separate validation set, the algorithm was always able to find a set of inputs and a network that was able to correctly predict each of the samples in the evaluation set. For example, training the networks with 38 samples and evaluating the GA on the

remaining set of 34 always resulted in a network/input set that could correctly classify all of the sample cases. However, when an additional validation set was added, results proved inconsistent. Even when the GA had maximized its fitness function by correctly classifying all of the samples in the evaluation set, the performance of the resultant network/input set was unpredictable.

Since these networks were never trained using the GA evaluation set and the network weights were re-initialized between each generation, overfitting through network training cannot be blamed for the variability in generalization abilities of the resultant networks. This result demonstrates that overfitting is an important problem for feature selection routines, particularly in domains as feature rich as gene expression.

3.1 VALIDATION

Golub (1999) divides the 72 AML/ALL samples into a training set of 38 and a test set of 34. Using this division, a simple class-mean/std filter and a weighted voting scheme⁵, they successfully predicted 29 of the 34 test samples. Lowering their decision criteria would have resulted in correct prediction of 32 of the 34. For the next series of runs, we used the same data and test set division.

Of the 38 samples in the training set, 4 were held out of network training. The entire set of 38 was used for the GA evaluation. Each run continued until a network/input set was discovered that could correctly classify all of the patterns in the evaluation set or until a maximum generation was reached. At this point, training ceased and the best network was evaluated on the 34 samples of the validation set.

In order to test the effectiveness of the GA, two trials were conducted. In the first, the GA was implemented as described above. In the second, the same algorithm and parameter settings were used except that fitnesses were randomized before reproduction.

With a population of 100 and the maximum number of generations set to 50, 500 runs with the GA produced 448 networks that succeeded in classifying 100% of the GA evaluation set. Of the remaining 42, all missed only 1 of the eval set. The GA produced an average of 5.5 errors on the validation set (of 34) with a standard deviation of +- 2.5 errors. (Removing the 42 runs that did not maximize the fitness function only improves the avg. validation error to 5.0).

⁵ This scheme is according to Dudoit *et al*(2000) diagonal linear discriminant analysis with the variance mistakenly replaced by a standard deviation.

The random feature search actually faired surprisingly well, producing an average of 7 errors on the validation set and successfully classifying all of the evaluation patterns in 122 of the 500 runs.

Table 1: AML/ALL Results

	Random search	GA search
Avg generation finished	43 +- 12	22 +- 14
Test set error rate	0.2	0.16
Avg # validation set errors	7.1 +- 3.4	5.5 +- 3.4
Avg # eval set errors	1.7 +- 0.8	0.08 +- 0.3
Avg # hidden units	5.3 +- 1.7	6.1 +- 1.1
Avg # inputs	5.4 +- 1.5	5.5 +- 1.3

Results of 500 runs with population 100, hid_mut_rate 0.4, num_in_mut_rate 0.001, input_mut_rate 0.3, max_num_in 8, min_num_in 3, max_num_hid 8 and max_gen 50.

Based on these results, the feature space for AML/ALL seems to be extremely rich in features useful for neural network training. One interesting result is that both genetic and random search consistently identified many of the same genes. In fact the top six genes most frequently used as inputs for successful networks were the same for both random and genetic search.

3.2 ENSEMBLES

Although training set performance was excellent, validation performance was inconsistent and unsatisfactory. One common method of improving the generalization performance of classifier systems is the creation of ensembles (Breiman 1994). In this technique, the predictions of multiple classifiers are combined in order to produce the class prediction for each sample in the test set. Of the various ways of combining the classifiers to form an ensemble, the simplest is to give each classifier a single vote for each prediction to be made. The final prediction is simply the class with the most votes.

3.2.1 RESULTS - RANDOMLY CONSTRUCTED ENSEMBLES

Initially, tests were conducted with ensembles created from the networks constructed via genetic and random search as presented in section 3.1. For these ensembles, although no specific attempt was made to insert diversity, the extremely stochastic processes of feature selection, network architecture selection, and network training resulted in substantially heterogeneous ensemble components. For this treatment, the 500 runs of random and genetic search were randomly divided

into 50 trials of 10 runs (networks) each. For each trial, the classifications were made by polling the predictions of each of the networks in the ensemble.

For the ensembles created from the genetic search runs, this brought the test set error rate down to 0.05 with an average of 1.7 +- 1.1 errors per run. The mean error rate for the ensembles created with random search was 0.07 with an average of 2.4 +- 1.96 errors per run. In addition to the improved error rate, the ensembles produced much more consistent results.

3.2.2 SMARTER ENSEMBLE CREATION

It is widely acknowledged that if the “ambiguity” of the component classifiers of an ensemble can be increased without reducing their individual error rates, the advantage from using an ensemble is increased (Krogh & Vedelsby 1995). In this case, ambiguity refers to the amount of disagreement between the component classifiers of an ensemble. The more a set of classifiers disagrees on the classifications to be made, the greater the advantage from combining them. This is because ensembles improve on individual classifiers by taking advantage of a wider variety of mappings from feature space to target space. If each component in an ensemble represented the same mapping, nothing would be gained. Thus, when constructing an ensemble, diversity of the component classifiers should be considered in addition to the generalization error of each component (Cunningham & Carney 2000).

One way to promote diversity among the classifiers of an ensemble is to use different subsets of the available features to construct each component classifier. Ho (1998) demonstrates that constructing component classifiers even from randomly selected features can improve the performance of ensembles significantly as compared to ensembles constructed from all of the features available. Guerra-Salcedo & Whitley (1999) employ the same basic approach as we presented in section 3.2.1, using a GA to search for good subsets of features to use as the components of an ensemble. They show some improvement over the random feature selection technique, particularly in domains with large⁶ numbers of features. Neither of these approaches explicitly selects for ambiguity in the component classifiers.

We found that by explicitly measuring the ambiguity of different ensembles and selecting those ensembles with the highest ambiguity, error rates could be slightly improved. Following the concept outlined in

⁶ The greatest number of available features in the datasets used in Guerra-Salcedo & Whitley (1999) was only 180.

Cunningham and Carney (2000), we developed the following measure of ambiguity:

$$A(e) = \sum_{x=0}^n D(x)^{-1} \quad (1)$$

where the ambiguity of an ensemble e as measured across a test set of n samples is equal to the sum of the inverse of the number of disagreements $D(x)$ between the classifiers in the ensemble for all $x \in n$.

To identify the strength of this measure in predicting the accuracy of the ensembles, 1000 ensembles were randomly constructed from 100 networks found via genetic search. Each classifier was constructed with 11 randomly selected networks and its error rate and ambiguity were measured.

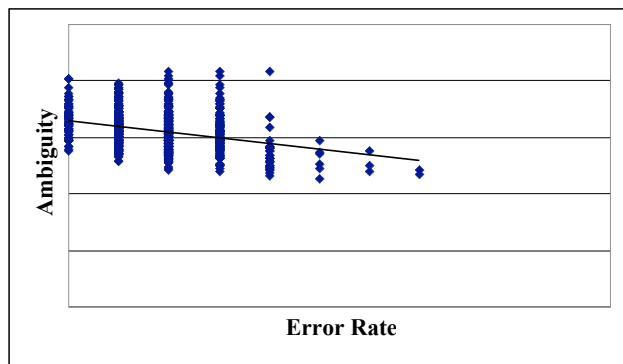


Figure 1: Ensemble error rate versus ambiguity as measured in equation (1). The line is derived from a linear regression.

Although we did not test the statistical significance of the relationship displayed in Figure 1, the results support the concept that ensemble accuracy can be improved through increasing component ambiguity and also that equation (1) seems to capture this relationship for our system.

The following table shows the advantage gained through ensemble formation and shows the correlation between ensemble ambiguity and ensemble generalization. Each column presents the results of using the same 100 networks/input sets discovered via genetic search but combining them in different ways.

Table 2: Comparing Ensemble Construction

	Single	Rand	Highest A	Lowest A
Number tested	100	1000	100	100
Number components	1	11	11	11
Avg. validation errors	5.9 +- 3.2	1.7 +- 0.99	1.5 +- 0.92	2.7 +- 1.3
Avg. eval errors	0.19 +- 0.4	0	0	0

In Table 2, the last two columns show the average number of errors for the 100 ensembles with the highest ambiguity and the 100 ensembles with the lowest ambiguity respectively. These were drawn from the same batch of 1000 ensembles used for the “Rand” column that displays the results of randomly chosen ensembles. The “Single” column shows the results from testing each of the networks used to create the ensembles individually.

These results suggest that the final stage of our algorithm should focus on the informed selection of ensemble components. Here, we randomly combined components and select those ensembles with the highest ambiguity measures. Although useful, this method could be substantially improved and optimized in future iterations of this work.

4 OTHER APPROACHES

Before concluding, it is important to point out the other methods that have been used for class prediction using gene expression data. Dudoit *et al* (2000) performed a comparison of various approaches to this problem. Focusing on the classification of tumor samples, they applied Fisher linear discriminant analysis (FLDA), diagonal linear discriminant analysis (DLDA), maximum likelihood discriminant rules, nearest neighbor classifiers, classification trees, and aggregation of classification trees, with additional tests employing bagging and boosting of the data.

For all of the datasets tested, they performed feature selection based on the ratio of each gene’s between-groups to within-groups sum of squares. Their classifiers employed either the top 30, 40, or 50 of these genes (with some tests with 200). Testing was conducted by training the classifiers on two thirds of the available samples and validating on the remaining third. They report that the nearest neighbor and DLDA predictors had the smallest error rates, while FLDA had

the highest. The error rates were generally surprisingly low across most methods. In fact, for the dataset that is used in the present study (AML/ALL), the mean error rate of DLDA was zero.

In order to compare our technique with those investigated by Dudoit and colleagues, we ran a series of trials composing our training and test sets using their 2:1 ratio and randomizing the samples in between runs.

Due to time constraints, only 21 networks were produced for each random sample set. This small number of runs did not provide sufficient variability to allow us to take advantage of the ambiguity measure. Hence, all 21 networks were combined to form the ensemble tested for each sample set. For 80 random sample set divisions, the average test error rate of the ensembles was 0.047. With 24 test cases, this corresponded to an average of 1.1375 errors with a standard deviation of 0.9.

While not obtaining the near perfect performance of DLDA, based on the figures presented in their paper, our algorithm seems to perform similarly to the nearest neighbor approach that was their second best classifier. Unfortunately, since they did not provide actual numbers for their error rates, it was not possible to do a direct comparison.

5 DISCUSSION

The good performance of straightforward methods such as DLDA might seem to discount the importance of the investigation of more complicated strategies such as neural networks and classification trees, however, as Dudoit *et al* point out, “there are factors other than accuracy which contribute to the merits of a given classifier”.

In the context of gene expression studies, the primary problem with techniques such as DLDA is that they ignore interactions between genes. These interactions are important not only because they could potentially aid in more complex classifications, but also because they are biological realities.

Genes do not act independently; they form complex genetic circuitry regulating specific biological functions. In order to understand the biological processes corresponding to different patterns of gene expression it is necessary to understand the relationships between the different genes in the system. If a classifier is to be useful in elucidating these potentially complex patterns of gene interaction, it is essential for it to be capable of representing those patterns.

It has already been established that neural networks can learn to recognize extremely complex patterns. As more training data comes available, they will undoubtedly be relied upon heavily in the classification and interpretation of gene expression studies. The primary challenge in using them will remain to be in feature selection and in deciphering the patterns that they discover.

6 CONCLUSIONS

To our knowledge, this paper represents the first successful application of neural networks to classification based on gene expression. It is also the first to apply the genetic algorithm wrapper methodology in a domain with thousands of available features. Investigations of the use of ensembles of neural networks show consistent improvement over single networks and add anecdotal support to the premise that increased component ambiguity increases the gain realized from ensemble formation. On this dataset, our method is not quite as accurate as the best linear method. Despite this, the GA/neural network approach remains a useful tool for comparison and provides a much greater potential for scaling to more complex classifications.

References

- Breiman L. (1994). Bagging Predictors. Technical Report 421, Dept. Statistics Technical Report 421, University of California, Berkeley, California.
- Chtioui Y., Bertrand D., Barba D., (1998). Feature Selection by a Genetic Algorithm. Application to Seed Discrimination by Artificial Vision. *J. Sci Food Agric* **76**: 77-86.
- Cunningham P., Carney J., (2000). Diversity versus Quality in Classification Ensembles Based on Feature Selection, Trinity College Dublin, Computer Science Technical Report: TCD-CS-2000-02, submitted to ECML-2000.
- Dudoit S., Fridlyand J., Speed T.P. (2000). Comparison of discrimination methods for the classification of tumors using gene expression data. Technical Report, Department of Statistics, UC-Berkeley
- Ho, T.K. (1998). The Random Subspace Method for Constructing Decision Forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 8, 832-844.
- Golub T.R., Slonim D.K., Tamayo P., Huard C., Gaasenbeek M., Mesirov J.P., Coller H., Loh M.L.,

Downing J.R., Caligiuri M.A., Bloomfield C.D., Lander E.S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* **286**: 531-537.

Guerra-Salcedo C., Whitley D. (1999). Genetic Approach to Feature Selection for Ensemble Creation. GECCO, 1999, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO 99)*. Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, Smith, eds., Morgan Kaufmann, 236-243.

Hallinan J., Jackway P. (1999). Simultaneous evolution of feature subset and neural classifier on high-dimensional data. *1999 Conference on Digital Image Computing and Applications (DICTA'99)*.

Krogh A, Vedelsby J (1995). Neural network ensembles, cross-validation and active learning, in: *Advances in Neural Information Processing Systems 9*, G. Tesauro, D. Touretzky and T. Leen, eds., MIT Press, 231-238.

Naef F, Lim D.A., Patil N, Magnasco M.O. (2001), "From features to expression: High-density oligonucleotide array analysis revisited", LANL e-print <http://xxx.lanl.gov/abs/physics/0102010>

Plaut D., Nowlan S., Hinton G.E. (1986). Experiments on learning by back propagation. Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams (1986). Learning internal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1: Foundations, pp. 318-362. Cambridge, MA: MIT Press. Reprinted in Anderson and Rosenfeld (1988).

Setiono, R., Liu, H., (1997) Neural-Network Feature Selector, *IEEE Transactions on Neural Networks*, **8**, (3), 654-662.

Siedlecki W., Sklansky J. (1989). A Note on Genetic Algorithms for Large-scale Feature Selection. *IEEE Transactions on Computers*, 10, 335-347.

Yang J., Honavar V. (1998). Feature Subset Selection Using a Genetic Algorithm. In: *Feature Extraction, Construction, and Subset Selection: A Data Mining Perspective*. Motoda, H. and Liu, H. (Ed.) New York: Kluwer.