Symbolic Discriminant Analysis for Mining Gene Expression Patterns

Jason H. Moore, Joel S. Parker, and Lance W. Hahn

Program in Human Genetics, Department of Molecular Physiology and Biophysics,
519 Light Hall, Vanderbilt University, Nashville, TN, USA 37232-0700
moore, parker, hahn@phg.mc.Vanderbilt.edu

Abstract. New laboratory technologies have made it possible to measure the expression levels of thousands of genes simultaneously in a particular cell or tissue. The challenge for computational biologists will be to develop methods that are able to identify subsets of gene expression variables that classify cells and tissues into meaningful clinical groups. Linear discriminant analysis is a popular multivariate statistical approach for classification of observations into groups. This is because the theory is well described and the method is easy to implement and interpret. However, an important limitation is that linear discriminant functions need to be pre-specified. To address this limitation and the limitation of linearity, we developed symbolic discriminant analysis (SDA) for the automatic selection of gene expression variables and discriminant functions that can take any form. We have implemented the genetic programming machine learning methodology for optimizing SDA in parallel on a Beowulf-style computer cluster.

1 Introduction

New laboratory technologies such as DNA microarrays [1], the serial analysis of gene expression (SAGE) [2], and protein mass spectrometry [3] have made it cost-effective and efficient to measure the relative expression levels of thousands of different genes in cells and tissues. The availability of massive amounts of gene expression information afforded by such technologies presents certain statistical and computational challenges to those hoping to use this information to improve our understanding of the initiation, progression, and severity of human diseases. Current statistical and computational methods such as linear discriminant analysis are simplistic and inadequate. Linear discriminant analysis [4] is a multivariate statistical classification procedure that linearly combines measurements on multiple explanatory variables into a single value or discriminant score that can be used to classify observations. This method is popular because there is a solid theoretical foundation [5] and it is easy to implement and interpret [6]. However, an important limitation is that the linear discriminant functions need to be pre-specified and only the coefficients for each linear predictor are estimated from the data. This limitation is not unique to linear discriminant analysis. Linear, polynomial, and logistic regression also require a pre-specified model [7].

Symbolic regression was developed by Koza [8] as a means of identifying regression equations that do not need to be pre-specified. Identification of optimal symbolic regression models is accomplished using the genetic programming machine learning methodology, an approach that is capable of generating creative solutions to a particular problem because it is not limited by human preconceptions about what the solution should look like [9]. Symbolic regression applications have included the discovery of trigonometric identities [8], econometric modeling [8], forecasting power demand [10], and modeling chemical process systems [11,12].

The goal of the present study was to develop a symbolic discriminant analysis approach by extending symbolic regression and parallel genetic programming to identify optimal linear or nonlinear discriminant functions and coefficients. We applied this new supervised classification method to identifying combinations of gene expression variables that differentiate acute myeloid leukemia (AML) from acute lymphoblastic leukemia (ALL) using the data collected and analyzed by Golub et al. [13].

2 A Review of Linear Discriminant Analysis

Sir Ronald Fisher developed linear discriminant analysis (LDA) as tool for classifying observations using information about multiple variables [4]. Consider the case in which there are two groups with n1 and n2 observations and k variables measured per observation. Fisher suggested forming linear combinations of measurements from multiple variables to generate a linear discriminant score (l) that takes the form

$$l_{ij} = \alpha_1 x_{ij1} + \alpha_2 x_{ij2} + \ldots + \alpha_k x_{ijk} \tag{1}$$

for the ith group and the jth observation in that group where each α is a coefficient and each x is an explanatory variable (e.g. gene expression variable). The goal of LDA is to find a linear combination of explanatory variables and coefficient estimates such that the difference between the distributions of linear discriminant scores for each group is maximized.

Classification of observations into one of the two groups requires a decision rule that is based on the linear discriminant score. For example, if l_{ij} ¿ l_o then assign the observation to one group and if $l_{ij} \leq l_o$ then assign the observation to the other group. When the prior probability that an observation belongs to one group is equal to the probability that it belongs to the other group, l_o can be defined as the median of the linear discriminant scores for both groups. When the prior probabilities are not equal, l_o is adjusted appropriately. Using this decision rule, the classification error for a particular discriminant function can be estimated from the observed data. When combined with cross-validation or independent data, the prediction error can be estimated as well.

3 Symbolic Discriminant Analysis

An obvious limitation of LDA is the need to pre-specify the linear discriminant function. Additionally, optimal classification of observations into groups may not be possible with a linear combination of explanatory variables. To address these limitations, we developed a method called symbolic discriminant analysis (SDA) that is able to identify the optimal functional form and coefficients of the discriminant function that may be linear or nonlinear. The SDA approach is inspired by the symbolic regression approach of Koza [8].

The goal of Koza's symbolic regression [8] is to identify a mathematical function or equation, in symbolic form, that fits a set of numerical data. Functionally, symbolic regression equations take the form of a computer program that accepts input values from one or more independent or explanatory variables and produces predictions for the dependent variable as output. These symbolic regression equations are represented as binary expression trees. Each binary expression tree is rooted with a mathematical function called a node. Each node or tree branch is connected to another node or ends with a connection to a terminal representing one of the independent variables. The number of vertical levels in the tree that contain at least one node is referred to as the tree depth. In general, increasing the number of levels in the tree increases the size and complexity of the symbolic regression equations.

The first step in implementing symbolic regression is to decide which mathematical functions (e.g. +, -, /, *, log, etc.) will comprise the function set. Additionally, the list of predictor or explanatory variables (e.g. X_1, X_2, X_3, etc.) that will comprise the terminal set must also be established. Random constants can be added to the terminal set to allow coefficients to be estimated. Once the function and terminal sets are defined, the next step is to identify the combination of functions and explanatory variables that minimize the sum of the squared differences between the output or predicted values and the observed values of the dependent variable. When the number of potential symbolic regression models is effectively infinite, machine learning methods must be employed (see Section 4).

With SDA, the binary expression tree or symbolic equation is used to generate symbolic discriminant scores for each observation in each group. Once the symbolic discriminant scores are generated, the classification error can be estimated as described in Section 2. In this study, the function set consisted of addition, subtraction, division (protected from division by zero), and multiplication. The terminal set consisted of approximately 7100 gene expression variables from the human leukemia dataset [13]. This number of explanatory variables presents an extraordinarily large search space.

4 Parallel Genetic Programming

Genetic or evolutionary algorithms, neural networks, case-based learning, rule induction, and analytic learning are some of the more popular paradigms in machine learning [14]. Genetic algorithms perform a beam or parallel search of the solution space that is analogous to the problem solving abilities of biological populations undergoing evolution by natural selection [15,16]. With this procedure, a randomly generated 'population' of solutions to a particular problem are generated and then evaluated for their 'fitness' or ability to solve the problem. The highest fit individuals or models in the population are selected and then undergo 'recombination' or exchanges of model pieces. Recombination generates variability among the solutions and is the key to the success of the beam search just as it is a key mechanism of evolution by natural selection. Following recombination, the models are reevaluated and the cycle of selection, recombination, and evaluation continues until an optimal solution is identified. A limitation of genetic algorithms is that the solutions or models must be represented by one-dimensional binary arrays or 'chromosomes'. Thus, these algorithms lack some flexibility. Koza [8] has addressed this by developing genetic programming which operates on computer programs rather than binary arrays. This advance opened the door for the optimization of problems ranging from robotics to electronic circuit discovery [9].

As with any machine learning methodology [14], genetic programming is not immune to stalling on local optima [17]. To address this issue, distributed or parallel approaches to genetic programming have been implemented [8,17,18]. Here, multiple genetic programming populations are generated each evolving semi-independently. At regular iterative intervals, the best solution obtained by each population is migrated to all other populations. Thus, if one population stalls on a local optimum, its search can be rekindled by the immigration of good solutions from other regions of the search space. Parallel genetic programs are conveniently implemented on parallel computer clusters where each population evolves on a single node or processor with migration occurring to other nodes via message-passing [19].

We used parallel genetic programming in the present study to optimize the selection of symbolic discriminant functions and gene expression variables. We implemented the parallel genetic programming by integrating the lil-gp software package [20] with the parallel virtual machine (PVM) message-passing library [19]. The parallel genetic program was run on two nodes of the VAnderbilt Multi-Processor Integrated Research Engine or VAMPIRE, a 55 node Beowulf-style parallel computer system running the Linux operating system. Each node has two Pentium III 600Mhz processors, 256 Mb RAM, a network card, and a 10 Gb hard drive. A total of two populations were used each consisting of 200 individuals. We allowed the genetic programs to run a total of 100 iterations with migration between each population every 25 iterations. A recombination frequency of 0.6 was used along with a mutation frequency of 0.02. We limited all binary expression trees to a node depth of six to prevent symbolic discriminant functions from becoming too large to interpret.

5 Data Analysis

Two independent human leukemia datasets were available for analysis [13]. We selected this dataset because previous class prediction methods have been applied with marginal success [13]. The first dataset consists of 38 acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) samples and was used to develop symbolic discriminant functions of the gene expression variables. We used a leave-one-out cross validation strategy with the parallel genetic programming to select symbolic discriminant functions with low classification and prediction errors. For each 37/38 of the data, we ran the parallel genetic program and selected the resulting model that minimized the misclassification rate and correctly predicted the class membership of the single observation left out of the training step. Optimal symbolic discriminant functions were then evaluated for their predictive ability using the independent dataset consisting of 34 AML and ALL samples. All of the approximately 7100 gene expression variables were available for possible inclusion in a model.

6 Results

We identified two 'near-perfect' symbolic discriminant functions that correctly classified 38 out of 38 (100%) leukemia samples in the training dataset and correctly predicted 33 out of 34 (97.1%) leukemia samples in the independent second dataset. Additionally, we identified 16 'very good' models that correctly classified 38 out of 38 (100%) in the training dataset and correctly predicted 32 out of 34 (94.1%) in the independent dataset. The first near-perfect symbolic discriminant function had four different gene expression variables while the second had just two. Each had different combinations of gene expression variables and different mathematical functions suggesting that there may be many subsets of gene expression variables that define leukemia type. For example, the first discriminant function had the form

$$X_{2555} * (X_{1153} + X_{2289} + X_{3193}) \tag{2}$$

while the second had the form

$$X_{1835} + X_{2546} \tag{3}$$

The binary expression trees for these two symbolic discriminant functions are illustrated in Figure 1 while boxplots of the discriminant scores are illustrated in Figure 2. For the first model (2), the genes identified were a testis-specific cDNA on chromosome 17q (X_{2555}), erythroid beta-spectrin (X_{1153}), adipsin (X_{2289}), and nucleoporin 98 (X_{3193}). For the second model (3), the genes identified were CD33 (X_{1835}) and Rho-E (X_{2546}). Most of these genes are biologically related to leukemia or leukemia progenitor cells. For example, the erythroid beta-spectrin protein is a major component of red blood cell membranes and is expressed during normal erythropoiesis. The adipsin gene is part of a chromosomal cluster of genes that is expressed during myeloid cell differentiation and the nucleoporin 98 gene is located at a chromosomal breakpoint that is associated with AML. The CD33 gene encodes a differentiation antigen of AML progenitor cells and is a very well-know pathological marker of AML. Thus, from approximately 7100 different gene expression variables, SDA identified several genes with direct biological relevance to human leukemia. Further, the level of prediction accuracy obtained with these two symbolic discriminant functions was nearly perfect and significantly better than the class prediction methods of Golub et al. [13].

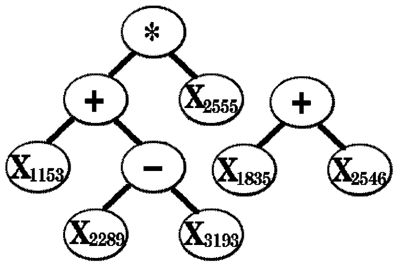

Fig. 1. Binary expression trees for model (1) on the left and model (2) on the right. Note that the nodes are arithmetic functions and the terminals are gene expression variables.

7 Discussion

This paper has presented symbolic discriminant analysis (SDA) for the identification and modeling of gene expression variables that can classify observations and predict clinical outcomes. This new approach to linear and nonlinear discriminant analysis is based on the building block idea of symbolic regression [8]. That is, we formed discriminant functions by selecting gene expression variables and mathematical functions of the gene expression variables. We implemented the genetic programming machine learning methodology [8,9,17] in parallel [18] to optimize selection of symbolic discriminant functions. Application of SDA to the problem of identifying symbolic discriminant functions of gene expression variables that classify and predict human leukemia types demonstrates the utility of this approach for both variable or feature selection and statistical modeling. This is particularly important for gene expression monitoring because as many as 10,000 or 15,000 gene expression variables are routinely being measured. Further, the relationship between different combinations of gene expression variables and clinical endpoints such as cancer type may be part of a very rugged fitness landscape.

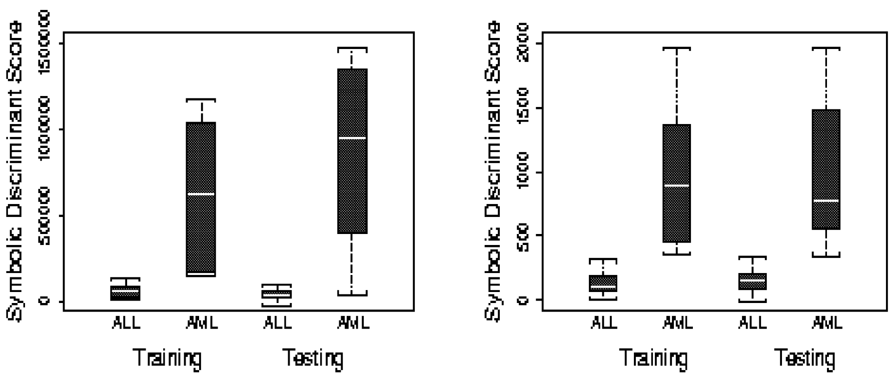

Fig. 2. Boxplots representing the distributions of symbolic discriminant scores generated using model (1) on the left and model (2) on the right. Boxplots are presented for the training or classification dataset (n=38) and the testing or prediction dataset (n=34). The shaded box of each plot represents the interquartile range of the distribution, the white line represents the median, and the whiskers or brackets above and below each box represent the maximum and minimum values, respectively. Note that for the training datasets, the whiskers of the ALL and AML scores do not overlap. This reflects the correct classification of every observation.

7.1 What Are the Advantages of SDA?

There are two important advantages of SDA over traditional multivariate methods such as linear discriminant analysis [4,5,6] and multiple logistic regression [21]. First, SDA does not pre-specify the functional form of the model. For example, with linear discriminant analysis, the discriminant function must take the form of equation (1). This limits the models to linear additive functions of the explanatory variables. With SDA, the basic mathematical building blocks are defined and then flexibly combined with explanatory variables to derive the best discriminant function. In this study, we used the four basic arithmetic functions as building blocks. However, other functions such as square root, logarithm, exponential, and absolute value could have been used. In fact, virtually any function can be added to the list with little modification of the code. We selected the basic arithmetic function in this study to help facilitate interpretation of discriminant functions. Additionally, combining multiple arithmetic functions can approximate many other mathematical functions such as square root and logarithm.

The second advantage of SDA is the automatic selection of variables from a list of thousands. Traditional model fitting involves stepwise procedures that enter a variable into the model and then keep it in the model if it has statistically significant marginal or independent main effect [7,21]. Interaction terms are only evaluated for those variables that are already in the model. This deals with the combinatorial problem of selecting variables, however, variables whose effects are mostly through interactions with other variables will be missed. This may be an unreasonable assumption for most complex biological systems. The SDA approach employs a parallel machine learning approach to selecting variables that permits interactions to be modeled in the absence of marginal effects. For example, the first 'near-perfect' model (2) identified from the leukemia dataset is a linear combination of interaction terms. No one variable contributes to the discriminant function independently of the others.

7.2 What Are the Disadvantages of SDA?

Although SDA has several important advantages over traditional multivariate statistical methods, there are several disadvantages. First, there is no guarantee that the genetic programming machine learning methodology will find the optimal solution. Heuristic searches tend to sacrifice finding an optimal solution in favor of tractability [14]. Implementing evolutionary-type machine learning algorithms in parallel certainly improves the chances of finding an optimal solution [8,17,18], but it is not a certainty. This is due to the stochastic nature of how genetic programming operates. The initial populations of solutions are randomly generated and the recombination and mutation occurs at random positions in the binary expression trees. Further, there may be a stochastic component to how the highest fit individuals are selected. For these reasons, evolutionary-type algorithms should be run multiple times with multiple parallel populations.

A second disadvantage of this approach is the computational requirement. Linear discriminant analysis can be performed on a standard desktop workstation while SDA requires a parallel computer cluster for optimal performance. When the number of genes to be evaluated is large (e.g. n=5,000 or n=10,000), the power of a parallel computer is required. Although such systems are fairly inexpensive to build, the time investment to establish and manage a parallel computing farm may be prohibitive to some.

A third disadvantage is the complexity of the symbolic discriminant functions obtained. An attractive feature of linear discriminant analysis is the simplicity of the models, which facilitates interpretation. Although the two near-perfect models obtained using SDA in the present study were fairly simple, SDA has the potential to generate rather large complex models. If so desired, this can be handled in several ways. One solution is to limit the size of the binary expression trees that the genetic program can build and manipulate. In this study, we limited the size of the symbolic discriminant functions by limiting the depth of the binary expression trees to a maximum of six. An alternative solution is to simplify the mathematical functions used. For example, by limiting the functions to addition and subtraction, SDA becomes linear discriminant analysis with heuristic variable selection.

7.3 Other Machine Learning Approaches to SDA

We selected parallel genetic programming as a machine learning methodology for optimizing SDA for several reasons. Most importantly, genetic programming provided the necessary flexibility to use binary expression tree representation of symbolic discriminant functions. Additionally, given the size and complexity of the solution space, performing multiple hill-climbing-type searches in parallel (i.e. a beam search) has a higher likelihood of success than a single hill-climbing search [14]. However, with this in mind, genetic programming is not the only option for optimizing SDA. For example, it might be advisable to carry out a limited exhaustive search as an initial step. The second near-perfect symbolic discriminant function (3) obtained during the analysis of the leukemia data was a simple additive linear function of two variables with no corresponding coefficients. This function could have been identified by carrying out an exhaustive search of the 25,201,450 possible pairwise combinations of gene expression variables. However, the first symbolic discriminant function (2), which contained a completely different set of genes with multiplicative effects, would have been missed.

Perhaps a more promising approach would be to combine the general search features of parallel genetic programming with the local search features of simulated annealing [22] or a stochastic search. For example, Krasnogor and Smith [23] employed a stochastic search once the genetic algorithm had reached a convergence state. Application of this new hybrid genetic algorithm seemed to improve optimization of the traveling salesman problem and a protein folding problem. We will explore the use of a combined local search for optimizing SDA in future studies.

8 Conclusions

We conclude from this study that the SDA approach provides a powerful alternative to traditional multivariate statistical methods for identifying gene expression patterns that are predictive of a clinical endpoint. The advantages of SDA include the ability to automatically identify important subsets of gene expression variables from among thousands of candidates and the ability to automatically identify the most appropriate mathematical functions relating the gene expression variables to a clinical endpoint. We anticipate this will be an important methodology to add to the repertoire of supervised pattern recognition and machine learning approaches for mining gene expression patterns.

References

1. Schena, M., Shalon, D., Davis, R.W., Brown, P.O.: Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science **270** (1995) 467–470
2. Velculesco, V.E., Zhang, L., Vogelstein, B., Kinzler, K.W.: Serial analysis of gene expression. Science **270** (1995) 484–487
3. Caprioli, R.M., Farmer, T.B., Gile, J.: Molecular imaging of biological samples: Localization of peptides and proteins using MALDI–TOF MS. Analyt. Chem. **69** (1997) 4751–4760
4. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Ann. Eugen. **7** (1936) 179–188

5. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis. Prentice Hall, Upper Saddle River (1998)
6. Huberty, C.J.: Applied Discriminant Analysis. John Wiley & Sons, Inc., New York Chichester Bisbane Toronto Singapore (1994)
7. Neter, J., Wasserman, W., Kutner, M.H.: Applied Linear Statistical Models, Regression, Analysis of Variance, and Experimental Designs. 3rd edn. Irwin, Homewood (1990)
8. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge London (1992).
9. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco (1999).
10. Lee, D.G., Lee, B.W., Chang, S.H.: Genetic Programming Model for Long–Term Forecasting of Electric Power Demand. Elec. Power Syst. Res. **40** (1997) 17–22
11. McKay, B., Willis, M., Barton, G.: Steady–State Modelling of Chemical Process Systems using Genetic Programming. Computers Chem. Engng. **21** (1997) 981–996
12. Willis, M., Hiden, H., Hinchliffe, M., McKay, B., Barton, G.W.: Systems Modelling using Genetic Programming. Computers Chem. Engng. **21** Suppl. (1997) S1161–S1166
13. 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.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science **286** (1999) 531–537
14. Langley, P.: Elements of Machine Learning. Morgan Kaufmann Publishers, Inc., San Francisco (1996)
15. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
16. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison–Wesley, Reading (1989)
17. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction. Morgan Kaufmann Publishers, San Francisco (1998)
18. Cantu–Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Boston (2000)
19. Leopold, C.: Parallel and Distributed Computing: A Survey of Models, Paradigms, and Approaches. John Wiley & Sons, Inc., New York (2001)
20. http://garage.cps.msu.edu/software/software–index.html
21. Hosmer, D.W., Lemeshow, S.: Applied Logistic Regression. John Wiley & Sons, Inc., New York (2000)
22. Kirkpatrick, S., Gelatt, C., and Vecchi, M.: Optimization by simulated annealing. Science **220** (1983) 671–680
23. Krasnogor, N., Smith, J.: A memetic algorithm with self–adaptive local search: TSP as a case study. In: Whitley, D., Goldberg, D., Cantu–Paz, E., Spector, L., Parmee, I., Beyer, H–G. (eds.): Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, Inc., San Francisco (2000)