Published in final edited form as:

Stat Appl Genet Mol Biol. 2004 ; 3: Article20. doi:10.2202/1544-6115.1038.

Hierarchical Bayesian Neural Network for Gene Expression Temporal Patterns*

Yulan Liang* and Arpad G. Kelemen†

* Department of Biostatistics, University at Buffalo, yliang@buffalo.edu

† Department of Computer and Information Science, Niagara University, Department of Biostatistics, University at Buffalo, akelemen@buffalo.edu

Abstract

There are several important issues to be addressed for gene expression temporal patterns' analysis: first, the correlation structure of multidimensional temporal data; second, the numerous sources of variations with existing high level noise; and last, gene expression mostly involves heterogeneous multiple dynamic patterns. We propose a Hierarchical Bayesian Neural Network model to account for the input correlations of time course gene array data. The variations in absolute gene expression levels and the noise can be estimated with the hierarchical Bayesian setting. The network parameters and the hyperparameters were simultaneously optimized with Monte Carlo Markov Chain simulation. Results show that the proposed model and algorithm can well capture the dynamic feature of gene expression temporal patterns despite the high noise levels, the highly correlated inputs, the overwhelming interactions, and other complex features typically present in microarray data. We test and demonstrate the proposed models with yeast cell cycle temporal data sets. The model performance of Hierarchical Bayesian Neural Network was compared to other popular machine learning methods such as Nearest Neighbor, Support Vector Machine, and Self Organized Map.

Keywords

Hierarchical Bayesian Neural Networks; Heterogeneous gene expression temporal patterns; Hyperprior; Monte Carlo Markov Chain

1 Introduction

cDNA microarray and other high throughput technologies provide thousands of gene expression measurements in parallel to uncover the secrets of life. The rich data environment has changed the standard research paradigm from a hypothesis driven approach to a data driven approach. In order to draw meaningful information from these data, wide ranges of statistical models, data mining, and pattern recognition tools have been applied and developed. These methods can be divided into two categories: differential gene expression and coordinated gene expression. The differential gene expression focuses on pair-wise comparisons between two conditions, such as normal versus diseased or mutant type versus wild type, and has been widely studied. The goal is to identify a target subset of significant genes that distinguish the observed

*We thank Dr. Richard Schmidt for his careful reading of this manuscript.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, bepress, which has been given certain exclusive rights by the author. *Statistical Applications in Genetics and Molecular Biology* is produced by The Berkeley Electronic Press (bepress). <http://www.bepress.com/sagmb>

phenotypes or cell types. Some recent literature about differential gene expression include Baldi and Long (2001), Tusher, Tibshirani, and Chu (2001), Pan (2002), and Storey and Tibshirani (2003). On the other hand, cells or phenotypes that appear similar may be functionally different or may follow a different clinical behavior, such as the response to drugs which at different experimental conditions can be dramatically different. This gave rise to the coordinated gene expression study, which assesses the expression levels of a large number of genes over a period of time or through a series of experimental conditions. The temporal information hidden in the gene matrix provides clues about the response of genes to the change in the environment and also to the gene activities at different conditions. Some examples of such time course experimental data are the cell cycle yeast data provided by Chu et al (1998), Cho et al (1998), and Eisen et al (1998). The characterization of multiple gene expression temporal patterns has become an important source of information in microarray research for biological process understanding, drug development, and efficient treatments therapies. The issues to be addressed in the coordinated gene study are the common functionalities, co-regulations, interactions, gene/protein networks, and pathways that ultimately are responsible for the patterns observed (Slonim, 2002 and Socci and Mitra, 1999). Pattern discovery tools, such as unsupervised learning with Hierarchical clustering, Self-organized map, Two-way clustering, and Principal Component Analysis drew a lot of attention (Eisen, et al, 1998, Alter, Brown, and Botstein, 2000, Hastiel, et al, 2000, Neal et al, 2000, and Yeung and Ruzzo, 2001). These approaches provide clues for genes that are related in their expression through linkage in a common developmental pathway.

Yet, there are some drawbacks of these clustering approaches. First, they may not detect genes with different functions showing similar profiles by chance or stochastic fluctuation from high level noise. Second, these measurements have not incorporated the temporal information, such as the features of significant nonlinear trend and the presence of sudden bursts of amplitude at irregular time intervals. Third, these methods may not detect indirectly correlated genes, which show weak correlations due to a time delay and nonlinear association. Last, they can not determine the number of patterns a priori automatically and the final partition of the data requires evaluation. Therefore, more hybrid and quantitative supervised models are needed to be developed in order to incorporate as much known biological and clinical information as possible. Dudoit, Fridlyand, and Speed (2002) have compared supervised linear discriminant approaches for the classification of tumors. Neural Networks (NN) are flexible nonlinear data driven models, which are easily modified and trained to follow any irregular or dynamic phenomena with nonlinear discrimination capability (Bishop, 1995; Haykin, 1999). The nonlinear dynamic stochastic behavior of the gene interactions can be monitored with neural networks with multiple hidden layers, nonlinear activation functions, and a correlation matrix, which can be randomly generated with a specified distribution. Neural networks also have a number of desirable properties, such as being tolerant to high level noise; high learning and adaptive capability on massively parallel data, and so on. These properties are well suited for time course microarray data features, such as irregularly time spaced, highly noised data. NNs have been applied for gene identification and cancer prediction, classification (Bicciato et al, 2001, Khan et al, 2001, Azuaje, 2001, Herrero, Valencia, and Dopazo, 2001). However, according to our knowledge, few works have been done on neural network application to multiple functional temporal patterns' classification/prediction.

Bayesian approaches are based on probability and the likelihood principle. They can overcome some model uncertainties and provide posterior probabilities of parameters. Furthermore, they can deal with missing data, such as unobserved time points and offer an easy way to estimate variations and high noise levels through a hierarchical Bayesian setting. Related studies of Bayesian approaches for microarray experiments include: Bayesian with regularized t-test (Baldi and Long, 2001), Bayesian regression analysis (West, 2000), non-parametric empirical Bayesian approach (Efron, et al., 2001), and parametric Bayesian model (Ibrahim, Chen, and

Gray, 2002). The intention of this work is to develop a novel approach, which integrates neural network techniques and Bayesian approaches in order to: (i) take into account different types of gene expression information and the features present in the data, including highly correlated inputs, temporal information, uncertainties, and noise estimation; (ii) address an issue of the correlation structure of multi-dimensional temporal data in multiple gene pattern studies.

The rest of the paper is organized as follows. In section 2 we describe the basic structure of gene expression classification with artificial neural networks. Furthermore, we develop a Hierarchical Bayesian Neural Network (HBNN) for both the independent and the correlated covariance structure of the temporal inputs. Also, the fully conditional posterior distributions for both the network parameters and hyperparameters and Monte Carlo Markov Chain (MCMC) simulation for sampling the conditional distribution in order to obtain marginal distributions of the parameters are discussed. In section 3 we demonstrate our developed method with a detailed analysis of time course microarray data sets, the effect of Bayesian learning, the convergence of MCMC, and the generalization performance. A comparison study of the proposed methods with popular approaches is also provided. Finally, we provide a brief discussion and conclusion in section 4.

2 Model Formulation

2.1 Neural Network Formulation

Since microarray data include massively parallel measurements with highly correlated inputs and overwhelming interactions, we consider a Feed-Forward Neural Network (FFNN) with a single hidden layer (a layer of neurons between the input layer and the output layer) architecture for modeling.

Let J be the number of input neurons in the network, which could correspond to the number of time points, or experimental conditions in temporal data. Let H and K be the number of hidden neurons and output neurons, respectively.

Data set $D = \{x_{ij}, y_i\}$, $i = 1, 2, \dots, n$; $j = 1, 2, \dots, J$ is given, where x_{ij} is the measurement of the i 'th gene at the j 'th time point. A row $\underline{x}_i$ is the input vector of an expression pattern of gene i . y_i is the target class, where the i 'th gene belongs and it takes a value from the finite set $\{0, 1, \dots, K-1\}$ of class labels.

We map gene expression pattern $\underline{x}_i$ to the network output $\hat{y}_i$ using:

$$\hat{y}_i = \Phi_0\left(\alpha_0 + \sum_{h=1}^H w_{h0} \Phi_h\left(\alpha_h + \sum_{j=1}^J w_{jh} x_{ij}\right)\right) \quad (1)$$

where Φ_h is the activation function from the input layer to the hidden layer, w_{jh} is the weight from the input layer to the hidden layer between neurons j and h , α_h is bias in the hidden layer, w_{h0} and α_0 are the weights and bias from hidden layer to output layer, Φ_0 is the activation function from the hidden layer to the output layer and is chosen to be the multinomial logistic function or soft function for multiple classes. Denote all the parameters as the weight vector $\underline{w} = (w_{jh}, w_{h0}, \alpha_0, \alpha_h)$ and the network architecture as N .

Define K latent values for each class, $u_{i0}, \dots, u_{iK-1}$:

$$u_{ik} = \alpha_0 + \sum_{h=1}^H w_{h0} \Phi_h\left(\alpha_h + \sum_{j=1}^J w_{jh} x_{ij}\right), \quad i=1, 2, \dots, n; k=0, 1, \dots, K-1$$

The class conditional probabilities $\hat{y}_{ik}$ that an input $\underline{x}_i$ belongs to class $y_i=k$ ($k=0, \dots, K-1$) given the network architecture N is given by:

$$\hat{y}_{ik} = P(y_i = k | x_i, w, N) = \frac{\exp(u_{ik})}{\sum_{k'=0}^{K-1} \exp(u_{ik'})} \quad (2)$$

The class prediction $\hat{y}_i$ of an observation x_i from K classes can be determined by the largest estimated conditional probability $\hat{y}_{ik}$ from K :

$$\hat{y}_i = C(x_i) = \arg \max_k P(y_i = k | x_i) \quad (3)$$

Due to the high level noise in gene expression data, repeating the sampling from the same data set may produce different target values with certain probabilities. A more specific goal of this work is through a Bayesian approach to estimate the noise levels, the associated network parameters, and the posterior probability $P(y_i = k | x_i, w, N)$, and then build a classifier with decision rules to make classifications and predictions. The assessment of the network performance from the learning stage is tested with data not seen before, which is called the testing data. This phase is often called generalization.

2.2 Hierarchical Bayesian Neural Networks for Gene Expression Classification

Our goal for constructing neural networks in the Bayesian framework for gene expression analysis is to attempt to recognize and categorize the complex temporal patterns with high uncertainties. There are two main sources of uncertainties when neural networks are applied for gene expression data. First, uncertainties may arise from the data itself, due to the experiment or biological noise. Second, they may arise from the modeling process, such as the estimated weights from different initial conditions or the deviation of multiple models based on the training results. Bayesian approaches are adequate due to their likelihood principle. On one hand, one can get estimates of variability of gene expression and noise probabilities. On the other hand, it allows one to do model averaging (Kass and Raftery, 1995) over several different fitted networks to improve predictive power and reliability of the results.

2.2.1 Hierarchical Bayesian Neural Networks with Independent Input Structure

—Note that eq. 1 is invariant when we shuffle the time points, which indicate that we have not incorporated the temporal information into the network model. Now we incorporate the temporal information with Bayesian prior setting into the neural network. We assume that the network parameters (weights from the input layer to the hidden layer and from the hidden layer to the output layer) are random variables with certain prior distributions, such as a Gaussian prior distribution. For a simple case, we assume that these parameters are independent with mean zero and common variance $1/\lambda$ (precision λ):

$$w | \lambda \sim \text{Gauss}(0, \frac{1}{\lambda}) \quad (4)$$

This simpler model has advantage in reducing the dimension of the parameter space and results in short computer run time. The precision λ (hyper-parameter) is the inverse of the square root of the variance and is associated with the noise level of each input to hidden weight. Since we have no specific knowledge of λ from the gene expression data, we used non-informative priors to acknowledge the high degree of uncertainty (noise levels) about the weights and allowed the data to guide model building in a higher degree. We assume λ is generated from vague prior Gamma distribution (which has a convenient conjugate prior) with mean α and shape parameter β .

$$P(\lambda) \sim \text{Gamma}(\alpha, \beta) \propto \lambda^{\frac{\beta}{2}-1} \exp(-\frac{\lambda\beta}{2\alpha}) \quad (5)$$

After the prior distributions of the network parameters and hyper-parameters have been setup, the posterior estimations of these parameters and hyper-parameters are updated according to the Bayes rule and the observed data:

$$P(w|D, \lambda, N) \propto P(D|w, \lambda, N) p(w|\lambda) \quad (6)$$

While the input-to-hidden weights connected to the same input have common prior λ , the posterior values are updated based on the data so that the irrelevant inputs (time points) have tighter distributions, which makes the corresponding weights take values close to zero with higher probability. Since a hyper-parameter is associated with each input (time), if the hyper-parameter for an input is small, the weight for that input will likely be small, and the hyper-parameters can be used to control the size of the weights associated with connections out of that input. This setting is useful for the automatic setup of the relevance of inputs from varying degree of correlated inputs to reduce the dimension of the parameter space and to automatically determine the model complexity (MacKay, 1995; Neal, 1997, 1998). This Bayesian setting is referred to as Automatic Relevance Determination (ARD), which helps to resolve traditional neural networks' and statistical models' multicollinearity problems (model can not set the coefficients of irrelevant variables to zero). These coefficients with nonzero values reduce the model's generalization performance.

One potential limitation of the above simple Bayesian setting is that the model assumption may be violated and result in poor performance. First, the prior distributions of random weights of the inputs are assumed to be uncorrelated with constant variance. We may achieve this through the standardization of the input x_j 's around their means, which can reduce dependence between weights and biases. Since gene expression data are usually fully balanced, we can achieve complete independence. Second, since we use the posterior distribution for our estimations, the assumed prior independence does not force the posterior distributions to be independent. Neural networks have special characteristics to solve the dependencies of the weights through the introduction of a hidden layer, in which the weights are aggregated via an activation function, and are adjusted simultaneously according to the learning algorithm.

2.2.2 Hierarchical Bayesian Neural Networks with Correlated Covariance

Structure—We may modify the assumption of independence to deal with the correlation structure of the multi-dimensionality of the time inputs. Uncorrelated weights with constant variance in an univariate distribution can be extended to the Multivariate Normal (MVN) distribution with correlated random weights with non-homogeneous covariance Σ , such as

$$w_i \equiv (\alpha_k, \alpha_h, w_{hk}, w_{jk})_i \sim MVN(w_0 \equiv (\alpha_k, \alpha_h, w_{hk}, w_{jk})_0, \Sigma), i=1, \dots, n \quad (7)$$

The prior distribution for w_0 , and Σ use conjugate distributions:

$$\begin{aligned} w_0 &\sim N(\mu, C) \\ \Sigma^{-1} &\sim W((\rho R)^{-1}, \rho), \end{aligned} \quad (8)$$

where W is Wishart distribution, which is a multivariate generalization of the Gamma distribution. $\rho \geq 2$ is scalar parameter of degree of freedom, and usually are chosen as small as possible to represent vague prior knowledge. Matrix R with $\rho \times \rho$ dimensions are specified to represent our prior guesses for weights and bias at the order of magnitude of the covariance matrix Σ^{-1} . Note that, if Σ is chosen to be constant diagonal, then it corresponds to the case we discussed in 2.2.1, independent weights and biases with constant variance.

Given the prior distribution, $w_i \sim MVN(w_0, \Sigma)$, due to the conjugate properties of the multivariate normal distribution, the fully conditional posterior distribution for the network weights can be given by a multivariate normal distribution:

$$w_i | x_i, w_0, \sum_{i=1}^{-1} N \sim MVN(B_i(X_i^T y_i + \sum_{i=1}^{-1} w_0), B_i) \quad (9)$$

$$\text{where } B_i^{-1} = (w_i - w_0)^T (w_i - w_0) + \sum_{i=1}^{-1}.$$

Marginalization (integration with respect to w_i) provides the posterior distribution for w_i , which is a multidimensional generalization of the univariate Student distribution (Webb, 2002).

Since $w_0 \sim N(\mu, C)$, and $p(w_0 | D, N) = \frac{p(D | w_0, N) p(w_0 | N)}{p(D | N)}$, the fully conditional posterior distribution for hyper-parameters w_0 can be given as:

$$w_0 | x, w, \sum_{i=1}^{-1} N \sim MVN(V(k \sum_{i=1}^{-1} \bar{w} + C^{-1} \mu), V) \quad (10)$$

$$\text{where } V = (k \Sigma^{-1} + C^{-1})^{-1} \text{ and } \bar{w} = \frac{1}{n} \sum_{i=1}^n w_i.$$

The fully conditional posterior distribution for Σ^{-1} follows the conjugate Wishart distribution:

$$\sum_{i=1}^{-1} x_i, w, w_0, N \sim W\left[\left(\sum_{i=1}^n (w_i - w_0)(w_i - w_0)^T + \rho R\right)^{-1}, n + \rho\right] \quad (11)$$

Sampling from Wishart distribution can be accomplished by simulation: if $x_1, x_2, \dots, x_v$ are

independent draws from $N_p(0, \Omega)$, then $V = \sum_{i=1}^v x_i x_i'$ is a $W(\Omega, v)$ random variate.

2.3 Making Inferences from HBNN and MCMC

Making inferences from the discussed HBNN model for gene classification includes several levels. The first (highest) level makes prediction and classification for new input data based on model comparison and averaging results from the training data. The second level makes inferences for the network parameters while we assume that the model structure is true, which is referred to as model fitting. The third level makes inference for hyper-parameters and their uncertainty. The last level repeats level 2 and level 3 with different initial conditions and network architectures and, based on the Bayesian evidence, it chooses the most probable model or uses weighted averaging (ensemble) of many models.

We are interested in finding the marginal posterior distribution of network parameters and the associated hyper-parameters in order to do classifications. The posterior joint distribution of the network weights w and hyperparameter λ is:

$$f(w, \lambda | D) = \frac{l(w, \lambda | D) \pi(w, \lambda)}{\int l(w, \lambda | D) \pi(w, \lambda) d\lambda dw} \quad (12)$$

where $\pi(w, \lambda)$ represents the joint prior for w and λ , $l(w, \lambda | D)$ is a likelihood function of λ and w for data D . Due to the high dimensional input space (the large number of parameters) and the non-independency between λ and w , the integrals of the joint distribution may not have an analytical solution. However, the fully conditional distributions of network parameters and hyper-parameters have conjugate properties at each level in the hierarchy and were given in 2.2. Therefore the Monte Carlo Markov Chain simulation or approximation is straightforward to be implemented in order to seek the marginal distribution for each parameter (MacKay, 1995; Neal, 1996, 1998). MCMC tries to obtain the posterior distribution of the network

parameter by simulating a Markov chain that has posterior as its equilibrium distribution. The prediction or classification for a new test case is made by averaging over the posterior distribution for all the parameters.

For the Monte Carlo method, Metropolis algorithms and Gibbs sampling can be implemented. In Gibbs sampling, the marginal distribution of weights can be sampled through sampling from the conditional distribution for each parameter when the other parameters are fixed. Since both the Gaussian distribution and the Gamma distribution have convenient conjugate properties, the fully conditional distributions have the same form as the conjugate priors. We can apply Gibbs sampling as follows. Choose arbitrary initial values $(\underline{w}^{(0)}, \lambda^{(0)})$, weights and parameters; then draw $\underline{w}^{(1)}$ from $f(\underline{w} | \lambda^{(0)}, D)$ and $\lambda^{(1)}$ from $f(\lambda | \underline{w}^{(0)}, D)$. After a large number of iterations, we obtain $(\underline{w}^{(K)}, \lambda^{(K)})$ ($K=B+1, \dots, B+M$), which is referred to as a Markov chain with ergodic properties. Hence it can converge to a unique desired distribution asymptotically.

Since we are interested in the marginal distributions of network parameters $\underline{w}$ and especially λ , good estimates for them are:

$$\begin{aligned} f(\underline{w}) &= \frac{1}{M} \sum_{k=1}^M f(\underline{w} | \sigma^{(B+k)}, \lambda^{(B+k)}, D) \\ f(\lambda) &= \frac{1}{M} \sum_{k=1}^M f(\lambda | \underline{w}^{(B+k)}, \sigma^{(B+k)}, D) \end{aligned} \quad (13)$$

Eq. 13 can be used to estimate parameters $E(\underline{w} | D)$, $Var(\underline{w} | D)$, $E(\lambda | D)$, $Var(\lambda | D)$, which can provide confidence intervals for all the parameters.

The difficulty with Gibbs sampling for the neural network parameters is that the sampling can be very slow to converge due to the high correlation of the high dimensional inputs and the high level noise, which is typically present in gene expression data. Moreover, using points selected at random from a simple distribution is not accurate because of the great variation in the posterior probability density in the high dimensional network space. The Metropolis algorithm could be more accurate, which produces a sequence of vectors and forms an ergodic Markov chain with a stationary distribution. However, it is still very slow to generate a candidate sample by randomly perturbing all weights from a high dimensional space. Also it has a hard time finding the desired direction, especially if applied to data with highly correlated inputs, such as gene expression data. To improve the Metropolis algorithms, we implemented the Hybrid Monte Carlo Markov Chain (HMC) method devised by Duane et al. (1987). HMC attempts to find the best direction by following the gradient with high probability, and following other directions with low probability. In HMC, the random walk aspect of the Metropolis algorithms is largely eliminated. It improves stability and the convergence of the algorithm, and speeds up the exploration of the parameter space. For hyper-parameters, Gibbs sampling was directly applied, which helped to minimize the amount of tuning for obtaining the best performance of HMC.

3 Application of HBNN on Yeast Cell Cycle Gene Temporal Data

3.1 Data Acquisition and Preprocessing

The transcriptional budding yeast microarray experiment, described by Chu et al. (1998) (<http://cmgm.stanford.edu/pbrown/sporulation/>), contains all yeast genes from diploid cells to haploid cells through the development of sporulation. The experiment contained 6118 genes, 477 of those with known classes (seven heterogeneous temporal patterns corresponding to different stages of development of sporulation) were identified by Chu's correlation based clustering method and has also been verified by Eisen's hierarchical clustering methods, Northern plots and other microarray data analysis (Chu, et al., 1998). These genes with known temporal patterns provide clues for the potential functions of thousands of uncharacterized

genes. Our goal is to build an effective classifier to characterize unknown genes with our discussed models using Chu's experimental information. Some genes with known classes of seven expression patterns provided by Chu et al. are shown in Table 1.

In Chu's experiment, genes that co-express in the same condition at different time points are of potential interest. The spotted cDNA microarrays containing *Saccharomyces cerevisiae* (yeast) were used to study the expression during meiosis and spore formation. mRNA samples were taken at seven time points: 0, 0.5, 2, 5, 7, 9, and 11.5 hours. For each time point, a red labeled cDNA pool was prepared. In addition for time 0 sample, a green labeled cDNA pool was prepared for reference. Seven microarrays were used in this study, one for each of the seven time points. Each array was probed with the green-labeled sample, mixed with one of seven red labeled samples. The original database contained four measurements for each spot: green signal, green background, red signal and red background. We used a log background-corrected ratio to normalize the data. The background-corrected ratio is given by the $(\text{red signal} - \text{red background}) / (\text{green signal} - \text{green background})$; log transformation was applied after the background-corrected ratio was computed. This normalization can well stabilize the variance and is more informative. More importantly, by doing this normalization, we can use a normal prior distribution and constant variance assumptions designed in 2.2.1. Figure 1 provides q-q normal plots and shows the effect of log background-correction for a randomly selected time point. The left figure shows that the assumption of normal distribution does not hold before normalization, but it does hold after it, which is shown in the right figure. Similarly, figure 2 shows the q-q plots for the other 6 time points after normalization. Since each of the slopes of the q-q plots provides the inverse of the variance and all the seven slopes are roughly the same, the constant variance assumption holds. To deal with the multiple heterogeneous distributed data and to get training data and testing data that contain genes proportional to the seven class sizes, we sorted the 477 genes according to their classes. After sorting, the data was proportionally split and randomly sampled according to the size of each class; a total of 157 samples (one third) were used as testing data and 320 samples (two third) as training data. After the data were partitioned, the training data were sent into the Hierarchical Bayesian Neural Network.

3.2 Implementation and Results

We implemented HBNN with independent weights and with correlated covariance matrix for gene temporal pattern classification. In the independent weights model the process of Bayesian learning uses a Feed-Forward Neural Network. Initial network weights and biases were randomly generated with common Gaussian prior distribution with mean 0 and variance $1/\lambda$. The hyper-parameter λ was generated from two hierarchical levels. The lower level hyper-parameter λ was generated from a Gamma distribution with a shape parameter 0.001 and with a mean given by a top level hyper-parameter common to all parameters of weights leading out from the input. This top-level hyper-parameter was generated from a Gamma distribution with a specified mean (0.1), and with yet another specified shape parameter (0.001). In the correlated covariance HBNN model, Bayesian learning also used a FFNN and Gaussian prior distribution for the weights, but the hyper-parameters of the correlated covariance matrix of the weights followed an Inverse Wishart distribution. The degree of freedom of the Inverse Wishart distribution was chosen to be the number of inputs, in this case 7. The scale matrix was chosen to be a 7×7 matrix, it was randomly generated and satisfied a positive semi-definite constraint.

Pure linear (purlin), tangent sigmoid (tansig), and log sigmoid (logsig) activation functions were tried as input to the hidden layer activation function as they were provided by the Matlab 6.0 software. The tangent sigmoid activation function on the hidden layer and the logistic function on the output layer gave a better performance than other activation functions. Several network architectures were tested by modifying the number of hidden neurons from 5 to 20.

12–15 neurons performed better than others. Batch mode learning was employed, in which the learning process updated the weights after the presentation of all training exemplars (i.e., after each epoch), and the weights were updated iteratively until the algorithm converged. Sensitivity analysis was performed through multiple test runs from random starting points to decrease the chance of getting trapped in a local minimum and to find stable results. For reliable results, each experiment was repeated 100 times, each with randomly generated initial weights and biases. MCMC was confined from 200 to 10,000 iterations in order to diagnose convergence, but in most cases the simulation curves did not jump significantly after 200 iterations, which meant convergence.

Since the network parameters have no significant meaning, the variations and the noise level estimations associated with each input (time point) is of potential interest. Figure 3 shows MCMC for hyper-parameters at seven time points when the independent input correlation structure is assumed. The x axis shows the number of iterations and the y axis shows the input-to-hidden hyper-parameter, which is the square root of the inverse of a variance. Here, each of the shown input-to-hidden hyper-parameter is a vector of parameters (weights) connecting a given input neuron to each of the hidden neurons. These hyper-parameters are used to control the magnitudes of weights out from each input. Therefore a high value of a given hyper-parameter signals a strong connection from an input to the hidden units, and probably a strong impact on the decision on the output layer for classification. Similarly, a low value of a given hyper-parameter signals a weak connection from an input to the hidden units, and probably a weak impact on the decision on the output layer for classification. We can see the effects of the ARD with a hyperprior Bayesian setting by looking at the change of the values of the input-to-hidden hyper-parameters over the course of learning. As it can be seen in the figure, time points 9 hours (set 5) and 2 hours (set 2) have strong relevance for the temporal patterns during training. Similarly, time points 30 minutes (set 1) and 11.5 hours (set 6) have low values, which indicate that they are less relevant for the temporal patterns. This shows that ARD with prior can be used to select the relevant inputs for model building.

In our first test, we chose three classes (Early I, Middle, and Early-Mid) with 221 genes as training data and 111 genes as testing data to test the model performance. The number of iterations we used was 200 to 10,000 and the number of hidden neurons was 15. We obtained the following outputs after 200 iterations and 150 runs: average log probability of targets: 0.202 ± 0.047 , fraction of guesses that were wrong: $8.11\% \pm 2.60\%$, correct classification rate: $91.89\% \pm 2.60\%$. These results showed that the proposed approach is very encouraging, even if the simple model was used.

Next, we increased the number of patterns to be classified to seven and 320 genes were used as training data and 157 as testing data. Using the simpler model for 15 hidden neurons, the correct classification rate for the testing data was $88.54\% \pm 0.097\%$ based, again, on 150 runs and 200 iterations each. The average log posterior probability of targets was -0.300 ± 0.039 . The fully correlated covariance matrix did not provide better results than the simple case (79%), which requires further investigation.

3.3 Effects of Automatic Relevance Determination

In order to visualize the effects of Automatic Relevance Determination for relevant input selection, we reduced the number of inputs to two. We analyzed two cases: In case I we used the two most relevant inputs (2 hours and 9 hours) provided by ARD. In case II we used the two least relevant inputs (0.5 hours and 11.5 hours). We randomly picked two classes, Early I and Early Middle, to demonstrate the effects. The Early I class contained 35 genes, while the Early Middle class contained 61 genes. We randomly chose 60 genes (20 from Early I and 40 from Early Middle) as training data and the rest, 36 genes, as testing data (15 from Early I and 21 from Early Middle). The correct classification rates from training and testing data for case

I and case II are summarized in Table 3. As shown in the table, the classification rates, based on the most relevant inputs (case I), provided much higher correct classification rates than the least relevant inputs (case II) for both training and testing.

3.4 MCMC with HBNN for Other Temporal Microarray Data

We tested the proposed models on two other yeast cell cycle data sets experimented by Cho's and Eisen's group and extracted the data from the Stanford genome research web site (<http://www.genome.stanford.edu>). For comparison purpose, we still included three output classes from Chu's data, and also three output classes for Cho's and Eisen's data. All these three data sets are subsets of the original data sets. Subsets were obtained such that classes with too few examples were omitted. In Cho's data, there are 17 inputs (time points), 280 genes with given three classes with known biological functions (patterns). Each gene measure is the normalized fluorescence between 0 and 160 minutes after cell cycle reinitiation. In Eisen's data, the gene matrix contained 79 time points over a variety of heterogeneous experimental conditions, which are important biological parameters. The data was generated from spotted arrays using samples collected at various time points during diauxic shift, mitotic cell division cycle, sporulation, temperature, reducing shocks, etc. The original data includes 2467 genes, each of which is associated with one of five functional classes. Table 4 is the summary of the data sets used for this study. All the data were standardized before they were fed into the neural network models. Table 5 shows classification results for the three data sets, all with the same number of output classes. The classification performance decreases as the number of inputs increases. This may indicate that, due to the increase in interactions among the inputs, ANN performance decreases. The performance of the methods may be improved by increasing the sample size despite the increase of noise. Figure 4 shows the MCMC simulation for Cho's data. Set 0 represents input-to-hidden hyper-parameter at time 0, set 1 at 30 minutes, and so on. Figure 5 provides similar information for Eisen's data. For better visualization we separated the 79 time inputs into three segments.

3.5 Comparison Study

For comparison purposes, we applied the Nearest Neighbor method with Mahalanobis distance measure for Chu's data in order to identify seven gene expression patterns. We partitioned the data into training and testing, the same way that we discussed earlier. The classification accuracy was 65.67% for seven patterns. In the Nearest Neighbor method, class 1 through class 6 was chosen as the learnable classes and class 7 as the control group. Implementation has been done in SAS 8.1. Results showed that class 5 was the best learnable class with the error rate of 9.15%, while class 1 was the worst learnable class with the error rate of 46.15%. These results are consistent with our proposed HBNN model where some classes were learned with a very high accuracy (90%), while others were not. This may be the reflection of insufficient training sample data for some classes, such as class 1, and a consequent lack of clear differentiation between them due to inherent uncertainties. In order to achieve high classification accuracy, we not only need sufficient sample size, but sufficient sample size of each class.

Other classification methods were also applied for the same data. Results of the Support Vector Machine with Adatron algorithm and the Self Organized Map with Euclidean distance metric were obtained with Neurosolutions 4.1. Support Vector Machine is a classification method for finding a hyperplane in a high dimensional space that separates training samples of each class while maximizing the minimum distance between the hyperplane and any training samples. SVM has properties to deal with high noise level. Brown et al. (2000) applied SVM to the budding yeast data produced by Eisen's lab (Eisen et al., 1998), where 79 time points with different experimental conditions with 2467 genes were associated with five functions

(classes). SVM worked well for that studied data. However, the best obtained correct classification rate for Chu's data was only 67.12%, in spite of the large number of initial settings for different parameters. The reason may be that Chu's data did not have sufficient samples for multiple heterogeneous classes. Self Organized Map is a winner-take-all architecture that maps the input space, using Euclidean distance metric, into the discrete output space. The classification accuracy of Self Organized Map for Chu's data was 71.12%. Again, the performance of these methods could be improved by increasing the amount of data despite the increase of noise.

4 Discussion and Conclusion

To characterize and classify heterogeneous multiple dynamic gene temporal patterns, we have developed and tested Hierarchical Bayesian Neural Network models with both an independent and a correlated covariance matrix. We demonstrated our models on a selected transcriptional profiling temporal data provided by a cDNA microarray experiment, and also other two yeast cell cycle data. The noise feature was well handled with the use of Bayesian techniques. With the Hierarchical Bayesian approach, the network parameters and the hyper-parameters were simultaneously optimized. Meanwhile, the Hierarchical Bayesian treatment helped to overcome the "black box" nature of neural networks, and provided us a more detailed inference of gene array data, such as the variations and noise estimations. In order to achieve the best construction of the model, appropriate data acquisition and preprocessing were performed, including normalization and randomization. Multiple iterated runs were also applied in order to evaluate the reliability of the results. Through the combination of the proposed models, the temporal patterns of gene expressions were well characterized, and the unknown genes were well classified into profiles with high classification accuracy. In most studies, the presence of noise in the data is considered undesirable, which should be eliminated. However, with the special features of HBNN, we estimated the noise levels and the variance of the models uniformly, provided the best construction of the models, and gained reliable classifications. Also, some components of such noise can be useful as additional information to capture complex and quantitative gene expressions in microarray data.

The accuracy of a classification method is influenced by the complexity of the dynamic patterns (decision boundaries), the heterogeneity of each class, the sample size of each class, the network complexity, and also is related to the corresponding data deduction and feature selection procedures. Data dimension reduction or feature selection helped to remove redundant features or noise and highlighted the most relevant features. Feature selection in our proposed model was done through Automatic Relevance Determination with hyperprior to learn the relevance of the different experimental variables to the output class. One advantage of our HBNN with ARD model is that it evaluates the performance of each variable subset on a trained target classifier, so we can optimize variable selection according to the performance of the target classifier. Therefore, ARD with hyperprior for automatic detection of relevant inputs for gene pattern temporal analysis could be useful, especially as preprocessors for highly correlated time inputs. Although our proposed model focuses on temporal pattern classification, it can also be applied for the classification of multiple class diseases, using the gene expression profile. In this case, the gene selection method would be highly related to the classification accuracy of multiple class diseases. This may be useful for a customized array design to automatically seek gene markers and important environmental factors, which have strong contributions to common diseases. All of these will be included in our further investigations.

Acknowledgements

Colophon

This work is supported in part by National Institute of General Medical Sciences grant (no. 5P20GM67650-02). Dr. Kelemen's work was also partially supported by the NYSTAR faculty development grant.

References

- Alter O, Brown PO, Botstein D. Singular value decomposition for genome-wide expression data processing and modeling. *PNAS* 2000;97(18):10101–10106. [PubMed: 10963673]
- Azuaje F. Unsupervised neural network for discovery of gene expression patterns in B-cell lymphoma. *Online Journal of Bioinformatics* 2001;1:26–41.
- Baldi P, Long AD. A Bayesian Framework for the Analysis of Microarray Expression Data: Regularized t-Test and Statistical Inferences of Gene Changes. *Bioinformatics* 2001;176:509–519. [PubMed: 11395427]
- Bicciato S, Pandin M, Didone G, Bello C. Analysis of an Associative Memory Neural Network for Pattern Identification in Gene expression data. *BIOKDD01, Workshop on Data mining in Bioinformatics*. 2001
- Bishop, CM. *Neural Network for Pattern Recognition*. Oxford: Clarendon Press; 1995.
- Brown MP, Grundy WN, Lin D, Cristianini N, Sugnet CW, Furey TSMA Jr, Haussler D. Knowledge-based analysis of microarray gene expression data by using supported vector machines. *PNAS USA* 2000;97(1):262–267. [PubMed: 10618406]
- Cho RJ, Campbell MJ, Winzler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell* 1998;2(1):65–73. [PubMed: 9702192]
- Chu S, DeRisi J, Eisen M, Mulholland J, Botstein D, Brown PO. The Transcriptional Program of Sporulation in Budding Yeast. *Science* 1998;282:699–705. [PubMed: 9784122]
- Duane S, Kennedy AD, Pendleton BJ, Roweth D. Hybrid Monte Carlo. *Physics letters B* 1987;195:216–222.
- Dudoit S, Fridlyand J, Speed T. Comparison of discrimination methods for the classification of tumors using gene expression data. *JASA* 2002;97(457):77–87.
- Efron B, Tibshirani R, Goss V, Chu G. Empirical Bayes Analysis of a Microarray Experiment. *JASA* 2001;96(456):1151–1160.
- Eisen MB, Spellman PT, Brown PO, Botstein D. Cluster analysis and display of genome-wide expression patterns. *PNAS* 1998;95(25):14863–14868. [PubMed: 9843981]
- Hastiel T, Tibshirani R, Eisen MB, Alizadeh A, Levy R, Staudt L, Chan WC, Botstein D, Brown PO. Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology* 2000;1(2):research 0003.1–0003.21.
- Haykin, S. *Neural Networks*. Prentice-Hall Inc; 1999.
- Herrero J, Valencia A, Dopazo JA. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics* 2001;17:126–136. [PubMed: 11238068]
- Ibrahim JG, Chen M, Gray RJ. Bayesian models for gene expression with microarray data. *JASA* 2002;97(457):88–99.
- Kass RE, Raftery AE. Bayes factors. *Journal of the American Statistical Association* 1995;90:377–395.
- Khan J, Wei JS, Ringner M, Saal LH, Ladanyi M, Westermann F, Berthold F, Schwab M, Antonescu CR, Peterson C, Meltzer PS. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Med* 2001;7(6):673–679. [PubMed: 11385503]
- MacKay DJC. Probable networks and plausible predictions -- a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems* 1995;6(3):469–505.
- Neal, RM. *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics No. 118. New York: Springer-Verlag; 1996.
- Neal, RM. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report No. 9702; Dept. of Statistics, University of Toronto. 1997.
- Neal, RM. Assessing relevance determination methods using DELVE. In: Bishop, CM., editor. *Neural Networks and Machine Learning*. Springer-Verlag; 1998. p. 97-129.

- Neal SH, Madhusmita M, Holter NS, Mitra M, Maritan A, Cieplak M, Banavar JR, Fedoroff VF. Fundamental patterns underlying gene expression profiles: Simplicity from complexity. PNAS 2000;97:8409–8414. [PubMed: 10890920]
- Pan W. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. Bioinformatics 2002;18(4):546–549. [PubMed: 12016052]
- Storey JD, Tibshirani R. Statistical significance for genomewide studies. PNAS 2003;100(16):9440–9445. [PubMed: 12883005]
- Slonim DK. From Patterns to Pathways: gene expression data analysis comes of age. Nature genetics supplement 2002;32:502.
- Socci, ND.; Mitra, P. Time series analysis of yeast *S. cerevisiae* cell cycle expression data. In: Bronberg-Bauer, E.; De Beucklaer, A.; Kummer, U.; Rost, U., editors. Proceedings of Workshop on computation of Biochemical Pathways and Genetic Networks; Heidelberg Germany. 1999.
- Tusher VG, Tibshirani R, Chu G. Significance analysis of microarrays applied to the ionizing radiation response. PNAS 2001;98(9):5116–5121. [PubMed: 11309499]
- Webb, A. Statistical Pattern Recognition. John Wiley & Sons, LTD; 2002.
- West, M. Bayesian regression analysis in the ‘Large p, Small n’ paradigm. Technical Report 00-22, Institute of Statistics and Decision Sciences; Duke University, CSE-2000-08-01. 2000.
- Yeung KY, Ruzzo WL. Principal component analysis for clustering gene expression data. Bioinformatics 2001;17:763–774. [PubMed: 11590094]2001

Figure 1. q-q normal plots before and after log background-correction of gene expression levels at time point 5 hours. The left figure with the curve shows that the normal assumption does not hold before the transformation. The right figure with the approximately straight line shows that the normal assumption holds after the transformation

Figure 2.

q-q normal plots of log background-corrected gene expression levels for 477 genes at time points 0, 0.5, 2, 7, 9, 11.5 hours, respectively. All six figures provide approximately straight lines, indicating that the normal prior distribution holds for each given time point. Also, the slopes of the lines are approximately the same, which indicates that the constant variance assumption holds.

Figure 3. MCMC simulation for the input-to-hidden hyper-parameter associated with noise level estimations of different time points. x-axis: number of iterations, y-axis: input-to-hidden hyper-parameter. Seven colors correspond to different MCMC simulation curves for seven time points for Chu's data

Figure 4. MCMC simulation for the input-to-hidden hyper-parameter associated with noise level estimations of different time points. (Cho's data). x-axis: number of iterations of MCMC. y-axis: input-to-hidden hyper-parameter at each time point. Each color corresponds to a hyper-parameter at each time point.

Figure 5. MCMC simulation for the input-to-hidden hyper-parameter associated with noise level estimations of different time points. (Eisen's data). x-axis: number of iterations of MCMC. y-

axis: input-to-hidden hyper-parameter at each time point. Each color corresponds to a hyper-parameter at each time point.

Table 1
Some genes with seven gene expression patterns in budding yeast data

Metabolic	Early I	Early II	Early Middle	Middle	Middle Late	Late
ACSI	ZIP1	AGA2	QR1	YSW1	CDC27	SPS100
PYC1	DMC1	KGD2	PDS1	SPR28	DT2	
SIP4	HOP1	YPT32	YBL078C	SPS2	DT1	
CAT2	IME2	MRD1	APC4	YLR227C		
YCR100C	YDR374	SPO16	KNR4	ORC3		
CAR1		NAB4	STU2	YLL005C		
		YPR192W				

Table 2

Testing results of HBNN for some selected genes for Chu's data

Gene name	2 hours	9 hours	Target (True) output	Network output
YGR179C	-0.74	1.24	Early-Mid	Early I
YBR231C	-0.72	1.31	Early-Mid	Early I
YPL034W	-0.54	-0.46	Early-Mid	Early-Mid
YOR214C	-0.27	-0.69	Early-Mid	Middle
YGR226C	-1.19	-1.08	Early-Mid	Early-Mid
YPL033C	-0.21	-0.92	Middle	Middle
YKR005C	-0.66	-0.77	Middle	Early-Mid
YDR273W	-0.03	-1.23	Middle	Middle
YHR014W	-1.23	-0.69	Early I	Early I

Table 3

Correct classification rates for training and testing data for case I and case II, for inputs selected by Automatic Relevance Determination for Chu's data.

Selected time points (case)	Correct classification rate (Training result)	Correct classification rate (Testing result)
2 and 9 hours (case I)	93.33%	91.67%
0.5 and 11.5 hours (case II)	78.33%	66.67%

Table 4

Extracted subsets of yeast data from Chu's, Cho's and Eisen's experiments

Data set	#Inputs	Selected output classes	Size of training/testing
Chu	7	Early II, Middle, Middle Late	200/114
Cho	17	Early G1, Late G1, S phase	185/95
Eisen	79	Ribosomal, Transcription, Secretion	250/126

Table 5

Misclassification rates of HBNN with ARD on various data sets based on 150 runs and 200 iterations

Data sets	#Inputs	#Outputs	Average log probability	Error rate
Chu	7	3	-0.204± 0.048	8.11%± 2.60%
Cho	17	3	-0.466± 0.068	20.00%±4.13%
Eisen	79	3	-0.706± 0.065	29.56%±3.63%