Journal of Biomedical Informatics 48 (2014) 114–121

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

A novel artificial neural network method for biomedical prediction based on matrix pseudo-inversion

Binghuang Cai [*], Xia Jiang

Department of Biomedical Informatics, School of Medicine, University of Pittsburgh, Pittsburgh, PA 15206-3701, USA

ARTICLE INFO

Article history:
Received 9 August 2013
Accepted 11 December 2013
Available online 18 December 2013

Keywords:
Biomedical prediction and classification
Neural networks
Matrix pseudo-inversion
Least Absolute Shrinkage and Selection Operator (LASSO)
Single Nucleotide Polymorphism (SNP)
Cancer

ABSTRACT

Biomedical prediction based on clinical and genome-wide data has become increasingly important in disease diagnosis and classification. To solve the prediction problem in an effective manner for the improvement of clinical care, we develop a novel Artificial Neural Network (ANN) method based on Matrix Pseudo-Inversion (MPI) for use in biomedical applications. The MPI-ANN is constructed as a three-layer (i.e., input, hidden, and output layers) feed-forward neural network, and the weights connecting the hidden and output layers are directly determined based on MPI without a lengthy learning iteration. The LASSO (Least Absolute Shrinkage and Selection Operator) method is also presented for comparative purposes. Single Nucleotide Polymorphism (SNP) simulated data and real breast cancer data are employed to validate the performance of the MPI-ANN method via 5-fold cross validation. Experimental results demonstrate the efficacy of the developed MPI-ANN for disease classification and prediction, in view of the significantly superior accuracy (i.e., the rate of correct predictions), as compared with LASSO. The results based on the real breast cancer data also show that the MPI-ANN has better performance than other machine learning methods (including support vector machine (SVM), logistic regression (LR), and an iterative ANN). In addition, experiments demonstrate that our MPI-ANN could be used for bio-marker selection as well.

1. Introduction and background

In the biomedical area, predicting clinical outcomes and diagnosing disease from available information, such as clinical and genetic evidence, is an important task for patient care and disease cure, especially for cancer applications [1,2]. Biomedical prediction problems are widely encountered in clinical applications including prognosis, diagnosis, and prediction of response to therapy. As information technology and medical equipment rapidly develop, more and more data, including clinical and genetic information, can be collected for medical utilization, which can increase the accuracy of biomedical prediction. However, as data become very large, especially genome-wide data, the processing of the data and the computation time for biomedical prediction is time-consuming and difficult. The biomedical prediction problem has been increasingly investigated by biomedical and informatics researchers [1–4]. This paper will address this challenging problem via developing an effective method and its algorithm.

The biomedical prediction problem can be mathematically described as follows. Given an N-sample training data set with elements defined as $\{\mathbf{X}_i,\mathbf{Y}_i\}_{i=1}^{N}$, where $\mathbf{X}_i = [x_{i1},x_{i2},\ldots,x_{im}]^{\mathrm{T}} \in \mathbf{R}^m$ (with m being the number of features/attributes) and $\mathbf{Y}_i = [y_{i1},y_{i2},\ldots,y_{in}]^{\mathrm{T}} \in \mathbf{R}^n$ (with n being the number of targets), the prediction problem is to discover the relation between $\mathbf{X}_i$ and $\mathbf{Y}_i$ and develop a model to describe such a relation, so that the output of the model, $\widehat{\mathbf{Y}}_i$, can be as close to the actual targets $\mathbf{Y}_i$ as possible. The problem can be described as

$$\mathbf{X}_i \to \mathbf{M} \to Y_i,\ \text{s.t.}|\mathbf{Y}_i - \widehat{\mathbf{Y}}_i| \to 0,\ i \in \{1,2,\ldots,N\}. \tag{1}$$

The discovered model $\mathbf{M}$ can then be used for the outcome prediction of a new observation $\widetilde{\mathbf{X}}$. This is important and useful for genetic prediction, clinical diagnosis, and disease classification. The problem is difficult to solve because biomedical data are often discontinuous, incomplete (values missing) and large-scale.

Different models and methods have been developed for the biomedical prediction problem [3,5–11]. For instance, a Bayesian approach using the logistic regression model was presented in [5] for cancer classification and prediction. The risk prediction of prostate cancer recurrence was investigated in [6] through regularized rank estimation in partly linear AFT (Accelerated Failure Time) models using high-dimensional gene and clinical data. An automatically derived class predictor was presented in [3] to determine the

* Corresponding author. Address: Department of Biomedical Informatics, School of Medicine, University of Pittsburgh, 5607 Baum Blvd., Pittsburgh, PA 15206-3701, USA. Fax: +1 4126245310.

E-mail addresses: bic9@pitt.edu, bhcai8@gmail.com (B. Cai), xij6@pitt.edu (X. Jiang).

1532-0464/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jbi.2013.12.009

class of new leukemia cases based on gene expression monitoring by DNA micro-arrays. An effective hybrid approach for selecting marker genes was developed in [7] for phenotype classification using micro-array gene expression data. A Bayesian network model for disease outbreak prediction was developed in [8].

Artificial Neural Networks (ANNs) are widely used in science and information technology due to their notable properties including parallelism, distributed storage, and adaptive self-learning capability [12–15]. They have also been utilized to solve biomedical problems, especially in the areas of classification and prediction [9–11]. For example, an artificial neural network, which was developed in [9] to determine whether breast cancer is present based on the age of the patient, mass shape, mass border, and mass density, achieved high predictive rates. A noise-injected neural network was designed in [10] for the classification of small-sample expression data for breast cancer patients. It demonstrated superior performance compared to the other methods tested. Another artificial neural network approach was used to reduce the number of gene signatures for the classification of breast cancer patients and the prediction of clinical outcomes, including the capability to accurately predict distant metastases [11]. However, all these ANN methods become very time-consuming as data become bigger, because the traditional learning method based on back-propagation algorithm is employed, and therefore they may not be applicable to practical biomedical prediction. A hybrid neural network and genetic algorithm method was applied to breast cancer detection in [16]; it used a genetic algorithm to determine the weights of the neural network (i.e., a multi-layer perceptron (MLP)). However, time-consuming iterations are still needed to get the weights for this hybrid ANN method.

Based on our previous work on weight determination of neural networks [13,14] and related work on ANN learning [17], we develop a Matrix-Pseudo-Inversion based Artificial Neural Network (MPI-ANN) for biomedical prediction. MPI-ANN is a feed-forward neural network with one input layer, one hidden layer and one output layer. Most importantly, MPI-ANN can directly determine the weights of the neural network in one step using pseudo-inversion without a traditional weight-updating iteration. For comparison purposes, LASSO (Least Absolute Shrinkage and Selection Operator) [18,19] is also presented for the biomedical prediction problem. Experimental results based on a set of simulated data sets and a real data set demonstrate the effectiveness and efficiency of the developed MPI-ANN method. Not only does MPI-ANN significantly outperform LASSO in terms of prediction accuracy for the simulated datasets, but it also demonstrates better performance in terms of statistical measurements and efficiency than other machine learning methods including support vector machine (SVM), logistic regression (LR) and an iterative ANN when analyzing a real breast cancer data.

The remainder of this paper is organized in four sections. Section 2 presents the MPI-ANN and the LASSO methods, and also introduces the experiment data sets and methods. In Section 3, experimental results are described and analyzed. A discussion appears in Section 4, and Section 5 concludes the paper with final remarks.

2. Methods

In this section, the MPI-ANN method is presented and developed for the biomedical prediction problem. The comparative method, LASSO, is also presented. Experimental data sets and the experimental method are finally introduced.

2.1. MPI-ANN

To solve the biomedical prediction problem, a feed-forward neural network is constructed according to the structure diagram shown in Fig. 1. The constructed MPI-ANN has three layers, i.e., the input layer, the hidden layer, and the output layer. The inputs to the neural network are the observed values of the features in the data set, while the outputs are the targets.

Specifically, in the input layer of MPI-ANN, the kth ($k = 1,2,\dots,m$) input of the MPI-ANN is the observed value of the kth feature. Each neuron of the input-layer uses a linear activation function $f(x) = x$; i.e., the values input into the neural network are directly passed to the hidden layer. Moreover, the hidden layer has p neurons, which employs a group of activation functions f_l ($l = 1,2,\dots,p$); i.e., the lth hidden-layer neuron adopts f_l as its activation function. Different kinds of non-regular functions [14,17] can be employed as the activation function of the hidden nodes. In this paper, based on a universal approximation theorem [20], the sigmoid function (i.e., $f(x) = 1/(1 + e^{-x})$) is employed for the MPI-ANN, since it is continuous (and thus differentiable), its derivative can be computed quickly, and it has a limited range (from 0 to 1, exclusive) [20]. The connecting weights between the input and hidden layers, $u_{kl} \in \mathbf{R}$ ($k = 1,2,\dots,m$; $l = 1,2,\dots,p$), and the biases of the neurons in the hidden layer, $b_l \in \mathbf{R}(l = 1,2,\dots,p)$, are randomly generated in any intervals of $\mathbf{R}$, since random choice of the input weights and hidden layer biases can exactly learn the training observations, make learning extremely fast, and produce good generalization performance according to [17,21]. Furthermore, the neurons in the output layer also use a linear activation function, and the inputs from the hidden layer are summed as the outputs of the neural network. The MPI-ANN can be considered as a kind of BP (Back Propagation) neural network; so it could use an error back-propagation algorithm [13,15,22] as its training law to determine the weights, $w_{lj} \in \mathbf{R}$ ($l = 1,2,\dots,p$; $j = 1,2,\dots,n$), between the hidden and output layers. To avoid the lengthy learning iteration of the traditional error back-propagation (BP) algorithm based on the gradient-descent method [13,15,22], we develop a matrix pseudo-inversion based weight direct determination method to determine such weights for biomedical prediction.

Mathematically, the MPI-ANN model can be formulated as

$$y_{ij} = \sum_{l=1}^{p} h_{il} w_{lj}, \tag{2}$$

where $i = 1,2,\dots,N$; $j = 1,2,\dots,n$ and

$$h_{il} = f_l\left(\sum_{k=1}^{m} u_{kl} x_{ik} + b_l\right), \tag{3}$$

is the output of the lth node of the hidden layer for the ith sample.

Based on matrix theory [23], the MPI-ANN model (2) can be expressed as the following matrix form.

$$\mathbf{Y} = \mathbf{HW}, \tag{4}$$

where

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nn} \end{bmatrix} \in \mathbf{R}^{N\times n}, \quad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1p} \\ h_{21} & h_{22} & \cdots & h_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Np} \end{bmatrix} \in \mathbf{R}^{N\times p},$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pn} \end{bmatrix} \in \mathbf{R}^{p\times n}.$$

The matrix-form error-function for MPI-ANN is expressed as follows:

$$E = \|\mathbf{Y} - \mathbf{Y}\|^2 = \|\mathbf{Y} - \mathbf{HW}\|^2, \tag{5}$$

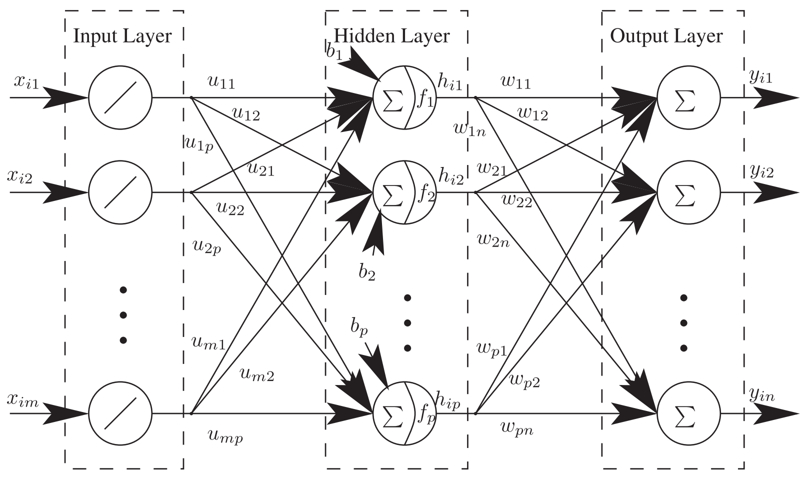

Fig. 1. Structure diagram for MPI-ANN.

which is equivalent to the traditional element-wise error function for ANN learning. Next, we present a matrix-based weight determination theorem based on previous work [13,14,17].

Theorem. *Given observation* $\{\mathbf{X},\mathbf{Y}\}$, *the weight matrix* $\mathbf{W}$ *of MPI-ANN can be directly determined by*

$$\mathbf{W}=\mathbf{H}^{\dagger}\mathbf{Y}, \tag{6}$$

where $\mathbf{H}^{\dagger}=\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}$ is the pseudo-inverse (Moore-Penrose generalized inverse) [23] of matrix $\mathbf{H}$ defined in (4) and (3).

Proof. Based on the variable definitions of the matrix-form MPI-ANN model (4) and the definition of the matrix-form error function (5), the traditional training BP algorithm [13,15,22] for the neural network can be written in matrix form as follows according to matrix theory [23]:

$$\begin{aligned}\mathbf{W}(k+1)&=\mathbf{W}(k)-\eta(\partial E/\partial\mathbf{W})|_{\mathbf{W}=\mathbf{W}(k)}\\&=\mathbf{W}(k)-\eta\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{W}(k)-\mathbf{Y}).\end{aligned} \tag{7}$$

After a sufficient number of iterations, the error function (5) would become small enough, i.e., $\lim_{k\to\infty}E=0$ and the weights can then be obtained. At the same time, the values of the connecting weights would become unchanged; i.e., $\lim_{k\to\infty}(\mathbf{W}(k+1)-\mathbf{W}(k))=0$. Thus, we can let $\mathbf{W}(k+1)=\mathbf{W}(k)=\mathbf{W}$. Eq. (7) then reduces to $\mathbf{H}^{\mathrm{T}}(\mathbf{HW}-\mathbf{Y})=0$ and we have $\mathbf{W}=(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y}$. As the pseudo-inverse (Moore-Penrose generalized inverse) [23] of matrix $\mathbf{H}$ can be defined as $\mathbf{H}^{\dagger}=(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$, we have $\mathbf{W}=\mathbf{H}^{\dagger}\mathbf{Y}$. The proof is now complete. □

In fact, the weight $\mathbf{W}$ determined by (9) is the least square solution of linear Eq. (6) based on an error function (5) [17]. We see that the weights can be determined by (6) in one step, which is much more efficient than traditional feed-forward neural network with lengthy training iterations. Preliminary research using simulations has shown that much less computational time (e.g., about 1850 times less time in the example of non-linear system identification in [14] and about 255 times less time in the example of real medical diagnosis in [17]) is required for this type of pseudo-inverse weight-determination ANN than conventional BP neural networks [13,14,17]. In all, we see the significantly superior efficiency of MPI-ANN over traditionally iterative ANN, which has been analyzed above and also mathematically and practically proven in the analysis and results in previous work [13,14,17] (including our previous work [13,14]).

Based on the theoretical analysis of the MPI-ANN model above, the procedure of the MPI-ANN algorithm can be described by the following steps.

(1) *Network construction*: construct MPI-ANN based on the network structure shown in Fig. 1.
(2) *Weight and bias generation*: randomly generate weights, u_{kl} ($k=1,2,\ldots,m$; $l=1,2,\ldots,p$), between the input and hidden layers, as well as the biases of the hidden neurons, $b_l(l=1,2,\ldots,p)$.
(3) *Weight determination*: calculate the weights, w_{lj} ($l=1,2,\ldots,p$; $j=1,2,\ldots,n$), between the hidden and output layers based on (6) and the training data sample $\{(\mathbf{X}_i,\mathbf{Y}_i)\}_{i=1}^{N}$.
(4) *Prediction testing*: calculate the predicted outcome for the testing data sample $\{\widetilde{\mathbf{X}}_i\}_{i=1}^{N}$ using (2) based on the determined weights w_{lj}.

2.2. LASSO

For comparative purposes, the LASSO (Least Absolute Shrinkage and Selection Operator) method is also presented to solve the prediction problem. LASSO is a shrinkage linear method for regression, which is simple and often provides an adequate and interpretable description of how the inputs affect the output [18,19].

LASSO fits the following linear model for the jth ($j=1,2,\ldots,n$) target of the biomedical problem (1),

$$\hat{y}_{ij}=b_0+b_1x_{i1}+b_2x_{i2}+\ldots+b_mx_{im}, \tag{8}$$

where $b_0,b_1,\ldots,b_m$ are the model parameters and can be obtained by

$$\min\sum_{i=1}^{N}(y_{ij}-\hat{y}_{ij})^2 \ \text{s.t.}\sum_{k=0}^{m}|b_k|\leqslant s, \tag{9}$$

with s being the bound turning parameter. LASSO could also be expressed as the equivalent Lagrangian form

$$\hat{\beta}=\arg\min_{\beta}\left(\gamma_j-\boldsymbol{\chi}\beta\right)^{\mathrm{T}}(\gamma_j-\boldsymbol{\chi}\beta)+\lambda\|\beta\|_1, \tag{10}$$

where $\gamma_j=[y_{1j},y_{2j},\ldots,y_{Nj}]^T\in\mathbf{R}^{N\times1}$, $\beta_j=[b_0,b_1,\ldots,b_m]^T\in\mathbf{R}^{(m+1)\times1}$, $\boldsymbol{\chi}=[\chi_1,\chi_2,\ldots,\chi_m]^T\in\mathbf{R}^{N\times m}$, with $\chi_i=[1,x_{1i},x_{2i},\ldots,x_{Ni}]^T\in\mathbf{R}^{N\times1}$, and $\lambda\geqslant0$ determines the amount of shrinkage. We see that the computation of the LASSO solution is a Quadratic Programming (QP) problem. The QP problem could be solved readily using the MATLAB routine "QUADPROG" [24], or solved preferably by using neural networks [25,26]. Note that when $\boldsymbol{s}$ is large enough (i.e., $\lambda = 0$), LASSO is multiple linear least squares regression; when $s \geqslant 0$ (i.e., $\lambda > 0$) is a smaller value, LASSO solutions are shrunken versions of the least squares estimates [19].

The computation of the entire path of the LASSO solution can be achieved based on Least Angle Regression (LAR) [27], which is intimately connected with LASSO. That is, LAR provides an extremely efficient algorithm for computing the entire LASSO path, which gives the entire path of LASSO solutions. Different packages have been developed for the computation of LASSO, such as "lasso4j" [28] in JAVA and "glmnet" [29] in R [30]. In our experiments, "lasso4j" is used.

2.3. Data sets

Two different data sets are employed for the validation of the performance of the developed MPI-ANN algorithm as well as the LASSO method. They are SNP (Single Nucleotide Polymorphism) simulated data sets [31] and the publicly available UCI-BCW [University of California, Irvine (UCI) Breast Cancer Wisconsin] real data set [32].

The SNP simulated data are computer-generated SNP data. They consist of 28,000 simulated data sets generated from 70 different genetic models of 2-SNP strict epistasis. The models were developed based on 70 different penetrance functions that define a probabilistic relationship between genotype and phenotype, which lead to different sensitivities between SNPs and diseases [31]. For example, according to Supplementary Table 1 in [31] and our findings using Bayesian network based methods in our previous studies [33–35], Models 55–59 have the weakest broad-sense heritability (0.01) and a minor allele frequency (0.2), which would have the lowest detection sensitivity between features and target. In contrast, Models 25–29 have the strongest broad-sense heritability (0.4) and a major allele frequency (0.4), which has the highest detection sensitivity. For each model, there are 4 different sample-sizes of data sets, i.e. 200, 400, 800, and 1600. For each sample-size in each model, 100 data sets are generated. Within each data set, there are 20 features ($F_0, F_1, \ldots, F_{19}$) with values being 0, 1, or 2 (corresponding to three different states of a SNP), and one disease class indicator with value being 0 or 1 indicating non-disease or having-disease. The numbers of non-disease and having-disease samples are the same in each data set. Note that, for a generated pair of epistatic SNP values (i.e., F_0 and F_1), a set of other 18 SNPs (i.e., $F_2 - F_{19}$) assigned random values was appended to simulate SNPs that are non-informative with respect to the disease status [31–35]. The first and the second features (i.e., F_0 and F_1) are the predictors for the disease.

The UCI-BCW real data set was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [32], and is available through the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29). The UCI-BCW data set has 683 samples with complete attribute values. Each sample has 10 features/attributes (i.e., F_0 – ID Number, F_1 – Clump Thickness, F_2 – Uniformity of Cell Size, F_3 – Uniformity of Cell Shape, F_4 – Marginal Adhesion, F_5 – Single Epithelial Cell Size, F_6 – Bare Nuclei, F_7 – Bland Chromatin, F_8 – Normal Nucleoli, F_9 – Mitoses) and one class feature with value being 2 (for benign tumor) or 4 (for malignant tumor). Note that, there are 444 benign tumor samples and 239 malignant tumor samples in the data set.

2.4. Experimental method

The MPI-ANN algorithm and LASSO algorithm were implemented in JAVA [36] with the matrix package "JAMA" [37] employed for matrix calculation in the algorithm, while all experiments of MPI-ANN and LASSO were conducted in the eclipse environment [38]. In the experiments, as a representative example, the weights between the input and hidden layers of MPI-ANN were randomly generated within the range of $(-1, 1)$ and the biases of the hidden nodes were randomly generated within the range of $(0,1)$. The number of hidden nodes of MPI-ANN was fixed to 10, i.e., $p = 10$, which is chosen based on experiment testing of different performances of MPI-ANN with different numbers of hidden nodes. The experiments were run on a computer with Intel® Core™ 2 Duo CPU E7600 3.06 GHz, 4.00 GB RAM.

Since the targets of the two data sets include two class types (non-disease and having-disease for SNP data, benign tumor and malignant tumor for UCI-BCW data), the number of outputs of MPI-ANN was set to two. The first output y_{i1} would output 1 if the ith sample belongs to the first class, otherwise, it would output -1. The same definition holds for y_{i2} relative to the second class. Specifically, for SNP simulated data, $y_{i1} = 1$ and $y_{i2} = -1$ are for class value equal to 1, while $y_{i1} = -1$ and $y_{i2} = 1$ are for class value equal to 0. For the UCI-BCW data set, $y_{i1} = 1$ and $y_{i2} = -1$ are for the benign tumor class, while $y_{i1} = -1$ and $y_{i2} = 1$ are for the malignant tumor class.

The *correct rate* was employed as the accuracy performance evaluation index, which is the proportion of the number of correctly predicted samples. In the SNP simulated data experiments, the average correct rate for all sample-size datasets in each model was calculated and compared for different feature combinations (e.g., F_0, F_1, $F_0 - F_1$, $F_0 - F_2$, $F_0 - F_3$, $F_0 - F_4$, and $F_0 - F_{19}$). Feature combinations were selected to test the performance of MPI-ANN for different sizes of feature sets. Predictors F_0 and/or F_1 were included in each combination in order to test the capability of MPI-ANN for predictor discovery. For the UCI-BCW real data set, the average correct rate of each combination of features was computed and compared. This list of feature combinations for testing was selected based on the performance of each feature. Each feature was first input into the MPI-ANN separately and the top-performing features were used to generate new combinations of features. In the UCI-BCW data experiment, each feature combination was tested 100 times to get the average correct rate. In addition, the

Table 1
Experimental results of MPI-ANN and LASSO for SNP simulated data.

Feature combination	Correct rate				Outperformance% MPI-ANN: LASSO		p-Value MPI-ANN: LASSO	
	MPI-ANN		LASSO					
	Training	Testing	Training	Testing	Training (%)	Testing (%)	Training	Testing
F_0	0.524998	0.499619	0.517434	0.499686	1.46	−0.01	3.03×10^{-3}	4.22×10^{-11}
F_1	0.524987	0.500023	0.517518	0.499695	1.44	−0.07	7.06×10^{-3}	5.10×10^{-15}
$F_0 - F_1$	0.676720	0.661922	0.528117	0.501470	28.09	31.94	2.20×10^{-7}	2.04×10^{-6}
$F_0 - F_2$	0.634639	0.603899	0.533506	0.500638	18.94	20.60	1.10×10^{-4}	2.02×10^{-3}
$F_0 - F_3$	0.606649	0.566907	0.538429	0.500508	12.66	13.25	7.46×10^{-7}	1.21×10^{-3}
$F_0 - F_4$	0.587628	0.542026	0.543084	0.500541	8.20	8.28	5.22×10^{-5}	1.90×10^{-2}
$F_0 - F_{19}$	0.558203	0.501313	0.584856	0.499962	−4.56	0.27	4.41×10^{-2}	4.32×10^{-3}

outperformance percent and the *p*-value based on the *t* statistic test were calculated in order to compare the performance of MPI-ANN and LASSO.

The 5-fold cross validation method [39] was used for each data set to get the correct rates of training and testing. Each dataset was divided into five parts based on the proportion between case and control of the original dataset, i.e., the proportion in each of the five parts were approximately the same as the one of the original dataset (i.e., 1:1 for SNP simulated data, and 444:239 for UCI-BCW real data). For each fold of cross validation, one part of the five parts of the dataset was selected as the testing data while the remaining four parts were considered as the training data. This was repeated for all five parts and the average value of the correct rate was generated for both training and testing.

To demonstrate the efficacy of MPI-ANN further for practical applications, SVM and LR were also compared with MPI-ANN using the UCI-BCW dataset. SVM and LR were implemented using "LIBLINEAR" [40] in MATLAB. We conducted experiments for the prediction with all features (i.e., $F_0 - F_9$) of the UCI-BCW dataset as the input of the MPI-ANN and compared the results with SVM and LR. Correct rate, sensitivity, specificity and computing time were used as the evaluation indices for their comparison. Note that, the sensitivity and specificity are statistical measures for the performance of a binary classification. Sensitivity measures the proportion of true positives that are correctly identified, while specificity measures the proportion of true negatives that are correctly identified. Classification should be both sensitive and specific as much as possible [16]. Moreover, to show the superior efficiency of MPI-ANN as compared to iterative ANN methods, we compared the outcome of MPI-ANN with the latest iterative ANN method (i.e., the hybrid genetic algorithm and neural network (hybrid GA-NN) method presented in [16]) using the UCI-BCW dataset.

3. Results

In this section, the experimental results are presented, described and compared for the SNP simulated data and the UCI-BCW real data.

3.1. Results of SNP simulated data

The experimental results for the SNP simulated data are shown in Fig. 2 and Table 1.

Fig. 2 shows that MPI-ANN obtains the best correct rate (red solid curve in Fig. 2(a and b)) for each model when using the data of Features F_0 and F_1 (the predictors in the SNP simulated data sets as mentioned in subsection 2.3) as the inputs of MPI-ANN, as compared with the other feature combinations, both in training and in testing. The simulation results (including the ones using other feature combinations, e.g., $F_0 - F_5$, $F_0 - F_6, \ldots, F_0 - F_{18}$, not

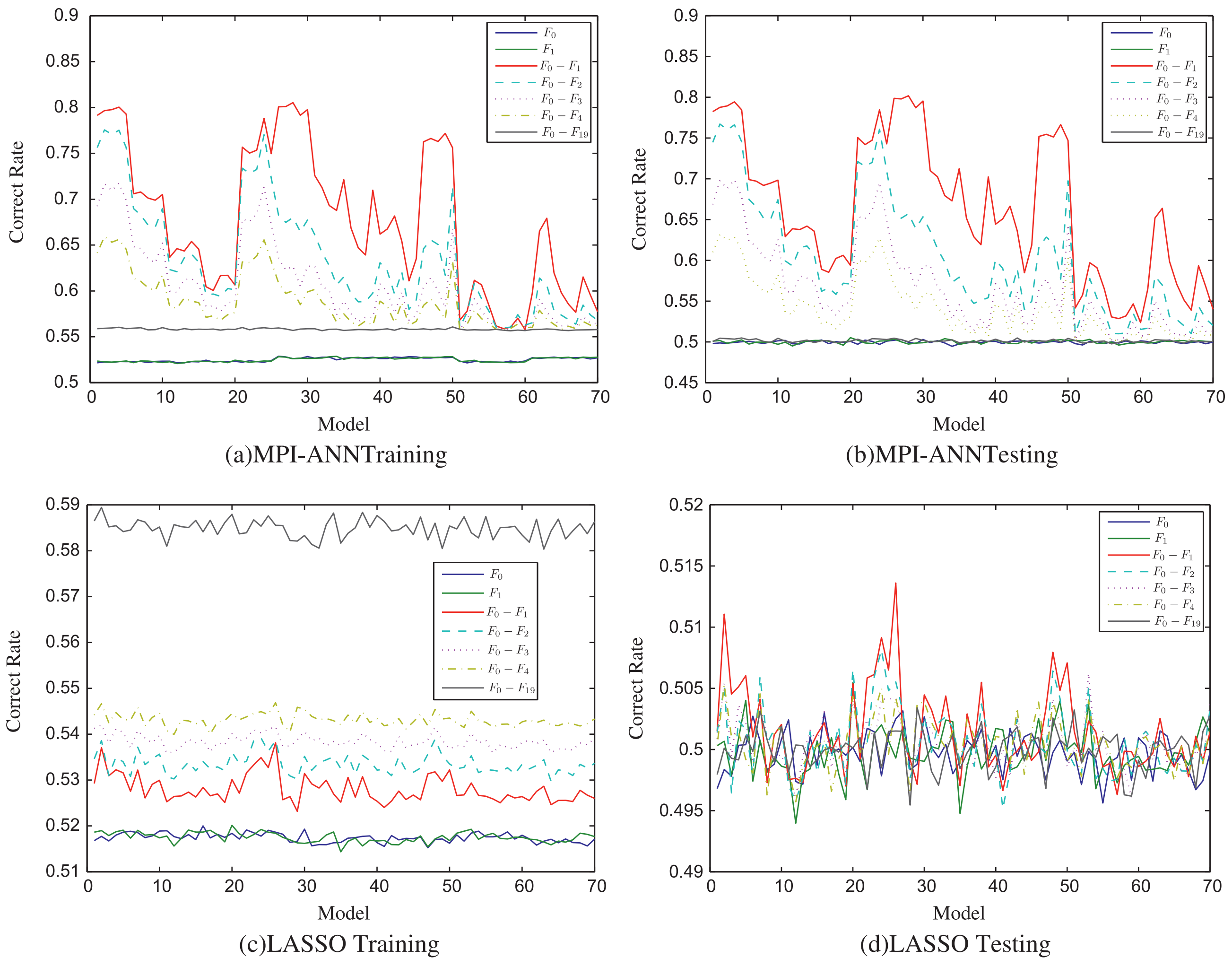

Fig. 2. Correct rate comparison of MPI-ANN and LASSO for SNP simulated data.

shown in Fig. 2) demonstrate that, when non-informative features are combined with the predictors (i.e., F_0 and F_1), a lower correct rate is achieved. Moreover, from Table 1, we see that the best average correct rates (i.e., 0.676720 and 0.661922) is obtained for training and testing when using Features F_0 and F_1. Meanwhile, the results from LASSO in the figure and table show that the best correct rate (less than 0.59) can only be achieved when all features (i.e., $F_0 - F_{19}$) are input for training, while other correct rates are around 0.5, especially for testing. This indicates that MPI-ANN can discover the predictors from the SNP simulated data, while LASSO may not have such capability for the SNP simulated data. This also implies that MPI-ANN can be applied in bio-marker selection in biomedical problems, e.g., via simply calculating the correct rates of different feature combinations and selecting the combination with best correct rate as the bio-marker.

Moreover, Fig. 2 also shows that MPI-ANN can perform well for both easy-to-detect models (e.g., Models 25–29) and hard-to-detect models (e.g., Models 55–59); i.e., it achieves higher correct rates for easy-to-detect models and lower correct rates for hard-to-detect models. This is consistent with our previous findings (e.g., the ones in [34]) of model-related sensitivity using Bayesian networks, which was mentioned in subsection 2.3. We see that MPI-ANN is able to discover how strong the features are related to the targets/diseases, while LASSO does not show this ability since LASSO exhibits no significant difference between correct rates for different models. The experiment results also show that MPI-ANN only needs less than 0.022 s to finish the training and less than 0.005 s to obtain the output for the testing data. The computational times are slightly larger than the ones for LASSO (around 0.001 s), but it is efficient enough for practical applications even in real-time situations.

Table 1 also shows that MPI-ANN outperforms LASSO in most of the feature combinations, especially when the combination is the predictors (i.e., F_0 and F_1). For example, MPI-ANN outperforms LASSO by 31.94% in testing and 28.09% in training when the combination is F_0 and F_1. The p-values based on the t-test that appear in Table 1 show that the superior performance of MPI-ANN over LASSO is significant in both training and testing, especially when the combination is F_0 and F_1. This result substantiates the superiority of MPI-ANN for biomedical prediction as compared with LASSO.

Above all, the experimental results based on SNP simulated data, show MPI-ANN's ability to detect biomarkers and discover features and target relationships, as well as MPI-ANN's significantly superior performance for biomedical prediction as compared with LASSO.

3.2. Results of UCI-BCW real data

The experimental results for the UCI-BCW real data are shown in Fig. 3. From the figure, we see that both MPI-ANN and LASSO can get high correct rates for the UCI-BCW data (from 0.75 to 0.95). However, MPI-ANN performs better (3.23% for training and 2.94% for testing) than LASSO for most combinations of features. The result of MPI-ANN outperforming LASSO is significant at the 9.26×10^{-10} level training and at the 2.90×10^{-9} level for testing. This again substantiates the effectiveness and superiority of the MPI-ANN method for biomedical prediction, as compared with LASSO. The experiment results also show that MPI-ANN uses less than 0.009 s for training and less than 0.003 s for testing, which is also efficient for practical applications. Note that the combinations of features are listed as follows:

$$\{\{F_1\}_1, \{F_2\}_2, \{F_3\}_3, \{F_4\}_4, \{F_5\}_5, \{F_6\}_6, \{F_7\}_7, \{F_8\}_8, \{F_9\}_9, \{F_2, F_3\}_{10}, \{F_2, F_5\}_{11}, \{F_2, F_7\}_{12}, \{F_3, F_5\}_{13}, \{F_3, F_6\}_{14}, \{F_3, F_7\}_{15}, \{F_5, F_6\}_{16}, \{F_5, F_7\}_{17}, \{F_6, F_7\}_{18}, \{F_3, F_5, F_6\}_{19}, \{F_2, F_3, F_5, F_6, F_7\}_{20}\}.$$

As mentioned in subsection 2.4, the feature selection for this list is based on the performance of each feature. Specifically, each of the ten features (i.e., $F_0 - F_9$) is input into the MPI-ANN separately, and the performances of the outcomes are compared. Top features are used to generate new combinations of features for testing. This shows the potential of using MPI-ANN as a tool for bio-marker selection via comparing the network outputs of different features.

The comparative results of MPI-ANN, SVM and LR for the prediction with all features (i.e., $F_0 - F_9$) in the UCI-BCW are shown in Table 2. From the table, we see that the correct rates of the predicted output of MPI-ANN are about 3–4% better than the ones of SVM and LR in both training and testing. The sensitivity values of MPI-ANN are about 4–5% better than the ones of SVM and LR. The specificity values for MPI-ANN are slightly better than the ones for SVM, and are almost the same as the ones for LR. Moreover, the training time of MPI-ANN is about 3.6 times less than that of LR and slightly less than that of SVM, while the testing time of MPI-ANN is more than 10 times less than those of SVM and LR. This shows the efficiency of MPI-ANN owing it using a one-step weight determination algorithm. In summary, MPI-ANN outperforms SVM

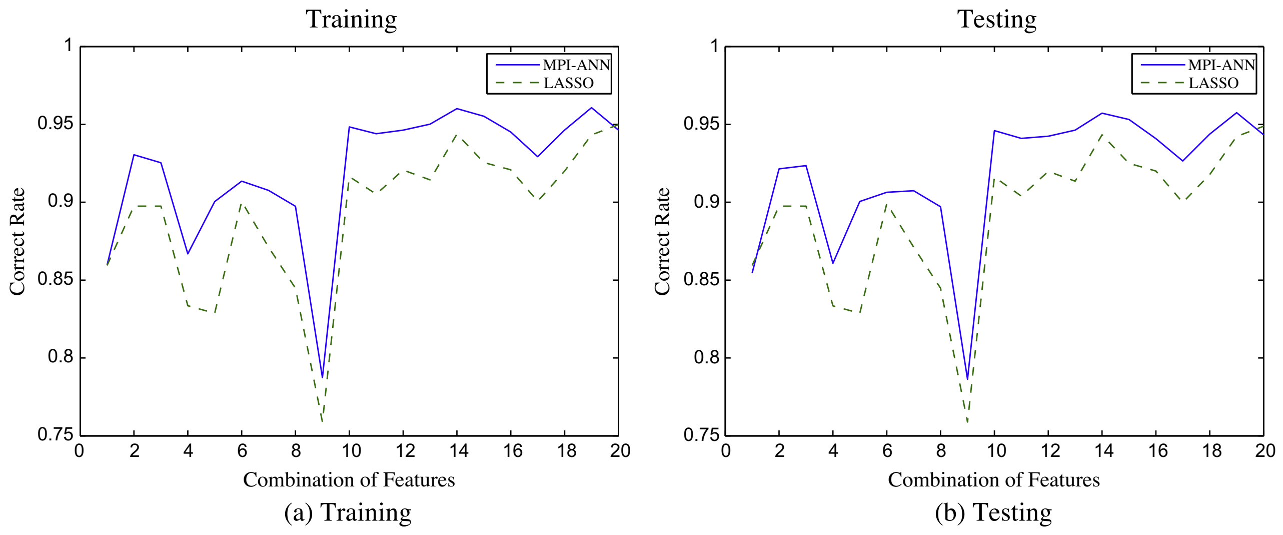

Fig. 3. Correct rate comparison of MPI-ANN and LASSO for UCI-BCW data.

Table 2
Comparison of MPI-ANN, SVM and LR for all-feature UCI-BCW data.

Methods	Correct rate		Sensitivity		Specificity		Computation time (s)	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MPI-ANN	0.902182	0.897344	0.930118	0.927830	0.850430	0.841355	0.012962	0.002720
SVM	0.872625	0.858051	0.889490	0.876864	0.840717	0.818350	0.015600	0.031200
LR	0.872990	0.857986	0.885083	0.865766	0.850403	0.847645	0.046800	0.031200

and LR for the real UCI-BCW dataset with all 9 features not only in statistical measurements but also in efficiency.

In addition, as compared with the latest ANN method (i.e., the hybrid GA-NN method in [16]) for the all-feature UCI-BCW dataset, the performance values of MPI-ANN in Table 2 are similar and comparable to the ones of the hybrid GA-NN method presented in the column "WPBC1" of Tables 3 and 4 of [16] (e.g., 0.9142 as of the best testing correct rate). However, the MPI-ANN takes much less time than the hybrid ANN method. The hybrid ANN takes at least 348 s (using a similar-speed computer with Intel® Core™ 2 Extreme CPU X9000, 2.80 GHz and 4.00 GB RAM for the experiments based on the same implementation language of JAVA) to get the results as shown in the column "WPBC1" of Tables 7 and 8 in [16], while MPI-ANN needs only 0.015682 s for both training and testing. That is more than 22,000 times less using similar-speed computer and same programming language. We see that MPI-ANN can achieve similar classification performance with the hybrid ANN method, but it takes much less computing time than the hybrid ANN method. Such superior efficiency is due to MPI-ANN's one-step weight determination rather than the iterations used by the hybrid GA-NN method.

In summary, the experiments based on the simulated data and the real data verify the effectiveness and efficiency of the developed MPI-ANN method for biomedical prediction, as well as the superior performance over LASSO and other machine learning methods.

4. Discussion

According to the results described in the previous section, the developed MPI-ANN performed well for both simulated data and real data in terms of accuracy and computation time. The results indicate that MPI-ANN is sufficiently effective and efficient to be applied in practical biomedical prediction applications. It is worth mentioning that MPI-ANN is readily applicable to other data sets with different numbers of features and targets via simply changing the numbers of input and output nodes. It can even handle data with targets including both discrete and continuous values. In addition, MPI-ANN can solve the linear classification problems for the simulated SNP data and the real breast cancer data more effectively than linear classifiers, such as, LASSO, SVM and LR. It is worth mentioning that MPI-ANN can also work well on non-linear classification problems due to its multiple-layer structure and changeable activation functions. In the future, we will apply MPI-ANN to non-linear classification problems.

For genome-wide data application, the number of features is huge (in general, more than ten thousands), which is much bigger than the numbers of features in the utilized data sets. Although MPI-ANN has shown its capability of processing a large number of data sets (28000 SNP simulated data sets), the handling of a huge number of features in a genome-wide data set using MPI-ANN is a challenging issue that should be considered in the future. Also, the efficiency of MPI-ANN for large scale biomedical problems compared to traditional methods (e.g., SVM, LASSO, LR, and iterative ANN) is the subject of future research. Fortunately, from the network structure and theory, MPI-ANN could easily scale to a large data set by adding input and output nodes for features and targets as mentioned above. Also, the experimental results demonstrated that MPI-ANN would be able to do feature selection, which would provide an effective way to process a huge number of features. Thus, future work could be the investigation of a feature selection algorithm based on MPI-ANN.

We set a fixed number of hidden nodes of the MPI-ANN for each data set in the experiments. From experiments, we have found that the number of hidden nodes may have some impact on the performance of the MPI-ANN. That is, by setting different numbers of hidden nodes, the accuracy performance of MPI-ANN may be different. Is there any relation between the number of hidden nodes and the outcome? Can we get an optimal number of hidden nodes? These are questions that need to be further investigated. Therefore, future research can concern the development of an optimal number determination algorithm for the number of MPI-ANN hidden nodes. In addition, whether multiple-hidden-layer ANN would improve the performance of MPI-ANN for biomedical prediction problem and how MPI-based weight determination method could be applied to the multiple-hidden-layer ANN are interesting topics that we will further investigate in the future.

5. Conclusions

We presented and developed an MPI-ANN (Artificial Neural Network based on Matrix Pseudo-Inversion) to solve the biomedical prediction problem based on clinical and genomic data in this paper. The weights of the MPI-ANN are directly determined without the lengthy learning iteration often used in the traditional neural network method, which is very inefficient for practical application. The LASSO (Least Absolute Shrinkage and Selection Operator) method has also been presented for comparison. We conducted experiments using SNP simulated data set and a breast cancer real data set. The results demonstrated the efficiency and the significantly superior performance of the MPI-ANN method for disease classification and prediction, as compared with LASSO and other machine learning techniques (e.g., SVM and logistic regression). The results also implied that the MPI-ANN could be employed for bio-marker selection. Future research may lie in testing micro-array genetic data, the development of a determination algorithm for the number of hidden nodes in MPI-ANN, and a feature selection algorithm based on MPI-ANN.

Acknowledgment

This work is funded by National Institutes of Health (NIH) under Grant LM010822.

References

[1] Shortliffe EH, Cimino JJ. Biomedical informatics: computer applications in health care and biomedicine. New York: Springer; 2006.
[2] Kim D, Shin H, Song YS, Kim JH. Synergistic effect of different levels of genomic data for cancer clinical outcome prediction. J Biomed Inform 2012;45:1191–8.
[3] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 1999;286:531–7.

[4] Cesario A, Marcus FB. Cancer systems biology, bioinformatics and medicine – research and clinical applications. New York: Springer; 2011.
[5] Zhou X, Liu KY, Wong STC. Cancer classification and prediction using logistic regression with Bayesian gene selection. J Biomed Inform 2004;37:249–59.
[6] Long Q, Moreno CS, Johnson BA. Risk prediction for prostate cancer recurrence through regularized estimation with simultaneous adjustment for nonlinear clinical effects. Ann Appl Stat 2011;5(3):2003–23.
[7] Wang Y, Makedon FS, Ford JC, Pearlman J. HykGene: a hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data. Bioinformatics 2005;21(8):1530–7.
[8] Jiang X, Cooper GF. A Bayesian spatio-temporal method for disease outbreak detection. J Am Med Inform Assoc 2010;17(4):462–71.
[9] Saritas I. Prediction of breast cancer using artificial neural networks. J Med Syst 2012;36:2901–7.
[10] Hua J, Lowey J, Xiong Z, Dougherty ER. Noise-injected neural networks show promise for use on small-sample expression data. BMC Bioinformatics 2006;7:274.
[11] Lancashire LJ, Powe DG, Reis-Filho JS, Rakha E, Lemetre C, Weigelt B, et al. A validated gene expression profile for detecting clinical outcome in breast cancer using artificial neural networks. Breast Cancer Res Treat 2010;120:83–93.
[12] Steriti RJ, Fiddy MA. Regularized image reconstruction using SVD and a neural network method for matrix inversion. IEEE Trans Signal Process 1993;41(10):3074–7.
[13] Zhang Y, Cai B. Artificial neural networks progress and debate during publishing process. Beijing: Publishing House of Electronics Industry; 2010.
[14] Zhang Y, Li W, Yi C, Chen K. A weights-directly-determined simple neural network for nonlinear system identification. In: Proceedings of IEEE international conference on fuzzy systems; 2008. p. 455–60.
[15] Rumelhart DE, McClelland JL, PDP Research Group. Parallel distributed processing. Cambridge (USA): MIT Press; 1986.
[16] Belciug S, Gorunescu F. A hybrid neural network/genetic algorithm applied to breast cancer detection and recurrence. Expert Syst 2013;30(3):243–54.
[17] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 2006;70(1–3):489–501.
[18] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. 2nd ed. New York: Springer; 2009.
[19] Hans C. Bayesian lasso regression. Biometrika 2009;96(4):835–45.
[20] Cybenko G. Approximations by superpositions of sigmoidal functions. Math Control Signals Syst 1989;2(4):303–14.
[21] Tamura S, Tateishi M. Capabilities of a four-layered feedforward neural network: four layers versus three. IEEE Trans Neural Networks 1997;8(2):251–5.
[22] Zhang Y, Li Z, Chen K, Cai B. Common nature of learning exemplified by BP and Hopfield neural networks for solving online a system of linear equations. In: Proceedings of IEEE international conference on networking, sensing and control; 2008. p. 832–6.
[23] Serre D. Matrices: theory and applications. New York: Springer; 2002.
[24] Optimization toolbox user's guide. Version 3.0.3. Natick (MA): The MathWorks Inc.; 2005.
[25] Cai B, Zhang Y. Different-level redundancy-resolution and its equivalent relationship analysis for robot manipulators using gradient-descent and Zhang et al.'s neural-dynamic methods. IEEE Trans Ind Electron 2012;59(8):3146–55.
[26] Cai B, Zhang Y. Bi-criteria optimal control of redundant robot manipulators using LVI-based primal-dual neural network. Optim Control Appl Methods 2010;31(3):213–29.
[27] Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression (with discussion). Ann Stat 2004;32(2):407–99.
[28] http://code.google.com/p/lasso4j/.
[29] http://cran.r-project.org/web/packages/glmnet/index.html.
[30] http://www.r-project.org/.
[31] Velez DR, White BC, Motsinger AA. A balanced accuracy function for epistasis modeling in imbalanced dataset using multifactor dimensionality reduction. Genet Epidemiol 2007;31:306–15.
[32] Mangasarian OL, Setiono R, Wolberg WH. Pattern recognition via linear programming: theory and application to medical diagnosis. In: Coleman TF, Li Y, editors. Large-scale numerical optimization. Philadelphia: SIAM Publications; 1990. p. 22–30.
[33] Jiang X, Barmada MM, Visweswaran S. Identifying genetic interactions in genome-wide data using Bayesian networks. Genet Epidemiol 2010;34(6):575–81.
[34] Jiang X, Barmada MM, Cooper GF, Becich MJ. A Bayesian method for evaluating and discovering disease loci associations. PLoS One 2011;6(8):e22075.
[35] Jiang X, Neapolitan RE, Barmada MM, Visweswaran S. Learning genetic epistasis using Bayesian network scoring criteria. BMC Bioinformatics 2011;12(89).
[36] http://www.java.com/en/.
[37] http://math.nist.gov/javanumerics/jama/.
[38] http://www.eclipse.org/.
[39] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceeding of the international joint conference on artificial intelligence; 1995.
[40] Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: a library for large linear classification. J Mach Learn Res 2008;9:1871–4. Software available at <http://www.csie.ntu.edu.tw/~cjlin/liblinear>.