Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Biomedical Informatics 37 (2004) 240–248
Journal of Biomedical Informatics
www.elsevier.com/locate/yjbin

Comprehensive vertical sample-based KNN/LSVM classification for gene expression analysis

Fei Pan[a,*], Baoying Wang[a], Xin Hu[b], William Perrizo[a]

[a] Department of Computer Science, North Dakota State University, Fargo, ND 58105, USA
[b] Laboratory of Structural Microbiology, The Rockefeller University, New York, NY 10021, USA

Received 14 June 2004

Abstract

Classification analysis of microarray gene expression data has been widely used to uncover biological features and to distinguish closely related cell types that often appear in the diagnosis of cancer. However, the number of dimensions of gene expression data is often very high, e.g., in the hundreds or thousands. Accurate and efficient classification of such high-dimensional data remains a contemporary challenge. In this paper, we propose a comprehensive vertical sample-based KNN/LSVM classification approach with weights optimized by genetic algorithms for high-dimensional data. Experiments on common gene expression datasets demonstrated that our approach can achieve high accuracy and efficiency at the same time. The improvement of speed is mainly related to the vertical data representation, P-tree,[1] and its optimized logical algebra. The high accuracy is due to the combination of a KNN majority voting approach and a local support vector machine approach that makes optimal decisions at the local level. As a result, our approach could be a powerful tool for high-dimensional gene expression data analysis.

Keywords: Data mining; *k*-Nearest neighbor; Support vector machine; Feature selection; P-tree; Gene expression; Machine learning

1. Introduction

The advent of whole genome-scale microarray gene expression experiment technologies opens new vistas in the area of analyzing various phenotype diseases, such as human cancers [1,2]. Classification approaches for gene expression data analysis, including classification decision tree, *k*-nearest neighbor classifier (KNN) [3], support vector machine (SVM) [4], neural network [5], etc., have been recognized as effective methods for distinguishing closely related cell types that often appear in the diagnosis of cancer.

However, classification analysis of microarray gene expression data often leads to neighborhood searches in a very high-dimensional data space. The publicly available datasets currently contain gene expression data with 5000–1000 genes on less than 100 observations and both numbers are expected to grow [6]. Classifying such high-dimensional data remains a contemporary challenge. In this paper, we propose a rapid and accurate classification approach, KNN/LSVM, for gene expression data analysis, which combines the KNN majority voting approach with local support vector machine approach to make optimal decisions at the local level. The most related subset of features is selected by the genetic algorithm using the weighted EIN-ring KNN as the fit function.

This approach is motivated from the experience of KDD Cup 2002, where we won an honorable mention by achieving the best score on broad problem, but not

* Corresponding author. Fax: +1 701 231 8255.
E-mail address: fei.pan@ndsu.nodak.edu (F. Pan).
[1] Patents are pending on the P-tree technology. This work is partially supported by GSA Grant ACT#:K96130308.

1532-0464/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2004.07.003

as accurate on narrow problem as broad problem [7]. The reason is that the data is high-dimensional and skew, with 3018 training instance on one class and 38 on the other of the narrow problem, which degrades the performance of the consensus voting approach. Using our KNN/LSVM approach with combination of majority voting and local boundary decision, we can improve the classification accuracy for gene expression analysis.

This paper is organized as follows. In Section 2, we first briefly review the recent related works in literature, and then the basic P-trees and the optimized operations are described in Section 3. In Section 4, we define a unique equal interval neighborhood rings, EIN-rings, and then present a new rapid accurate classification approach for microarray analysis. Finally, an accuracy and efficiency performance study is reported in Section 5 and we conclude the paper in Section 6.

2. Related works

Many methods have been used in cancer classification using gene expression, such as artificial neural network [5], support-vector machines (SVM) [4], *k*-nearest neighbors [3], and Golub et al. [2] classifier. Golub et al. [2] first employed a binary-class classifier, which is based on the consensus voting using correlation coefficients automatically discovered the distinction between the acute myeloid leukemia and the acute lymphoblastic leukemia. Furey [4] applied SVM to micoarray data that consists of both the classification of tissue samples, and an exploration of the data for mislabeled or questionable tissue results. Another study of SVM illustrated the method for predicting functional roles of 2467 uncharacterized genes from yeast *Saccharomyces cerevisiae* on the basis of expression data from DNA microarray hybridization experiments [8].

Nearest neighborhood classification approaches, also referred to as instance-based learning algorithms [9], are sometimes called "lazy" learning methods because they delay processing of classification until a new instance requests to be classified [1]. The key advantage of such learning approaches is that the classification decisions are made locally and differently for each new instance to be classified [10]. Cover and Hart [11] first studied the nearest neighbor approach and proposed the nearest neighbor classification rules. Wagner et al. studied the convergence of the nearest neighbor classification approach [12,13]. The property of constancy and convergence of nearest neighbor density estimates has been well studied theoretically by Moore, Devroye, and coworkers [14,15].

Feature selection is one of the crucial steps for a comprehensive classifier. Many approaches, such as principal component analysis, linear discriminant analysis, projection pursuit, have been proposed to reduce the dimensions of gene expression data and choose informative subset [16]. Golub et al. [2] employed a leave-one-out crossvalidation method to select a subset gene before classifying acute myeloid leukemia and acute lymphoblastic leukemia using neighborhood analysis. Feature selection using genetic algorithms is a multivariate approach, which is capable of selecting a subset of genes that are uncorrelated with each other. Li and Ooi both reported that they successfully applied GA to the gene selection to improve the accuracy of classification analysis [3,17].

3. Review of P-tree technology

The P-tree technology was initially developed by the DataSURG research group in North Dakota State University for spatial data [18,19]. The basic data structure for this technology is the Peano Count Tree (P-tree). P-trees are tree-like data structures that store numeric relational data in compressed format by splitting vertically each attribute into bits, grouping bits in each bit position, and representing each bit group by a P-tree. P-trees provide a lot of information and are structured to facilitate data mining processes. In this section, we briefly review the useful features of P-trees and propose optimized P-tree logical operations.

3.1. Data representation

We organize the gene expression data as a relational table with column of genes and row of experiments, phenotypes, or cell lines. Instead of using double precision float numbers with a mantissa and exponent represented in complement of two, we partition the data space of gene expression into intervals. This will enable us to work on a high-dimensional data by approaching the speed of the binary representation and achieving fine accuracy.

First, we need to decide the number of intervals and specify the range of each interval. For example, we could partition the gene expression data space into 256 intervals along each dimension equally. After that, we replace each gene value within the interval by a string, and use strings from 00000000 to 11111111 to represent the 256 intervals. The length of the bit string is base two logarithm of the number of intervals. The optimal number of interval and their ranges depend on the size of datasets and accuracy requirements, which could be determined by domain experts or preliminary performance experiments.

Suppose a datum with d features (genes), $X = (G_1, G_2, \ldots, G_d)$, and the binary representation of jth feature G_j as $b_{j,m}b_{j,m-1}\cdots b_{j,i}\cdots b_{j,1}b_{j,0}$, we decompose each feature into bit files, one file for each bit position. To build

a P-tree, a bit file is recursively partitioned into halves and each half into sub-halves until the sub-half is pure (entirely 1-bits or entirely 0-bits), or the length of sub-half is less than the minimum bound. The detailed construction of P-trees is illustrated by an example in Fig. 1.

For simplicity, the data with one feature attribute are shown in Fig. 1A. We represent the attribute as binary values, e.g., $(7)_{10} = (111)_2$. Then vertically decompose them into three separate bit files, one file for each bit position, as shown in panel B. The corresponding basic P-trees, P_1, P_2, and P_3, are constructed from the three bit vectors correspondingly by recursive partition with minimum bound of length one, which are shown in panels C, D, and E. As shown in Fig. 1C, the root of P_1 tree is 3, which is the 1-bit count of the entire bit file. The second level of P_1 contains the 1-bit counts of the two halves, 0 and 3. Since the first half is pure, there is no need to partition it. The second half is further partitioned recursively.

AND, OR, and NOT logic operations are the most frequently used P-tree operations. We use $\wedge$, $\vee$, and prime ($'$) to denote P-tree operations AND, OR, and NOT, respectively. We define a basic predicate tree called Pure-1 trees (P1-trees) for efficient operation. A node in a P1-tree is "1" if and only if that sub-half is entirely 1-bit. Fig. 2 shows the P1-trees corresponding to P_1, P_2, and P_3 in Fig. 1.

The P-tree logic operations are performed level-by-level starting from the root level. They are commutative and distributive, as they are simply pruned bit-by-bit operations. For instance, ANDing a pure-0 node with any results in a pure-0 node, ORing a pure-1 node with any results in a pure-1 node. In Fig. 3, (A) is the result of $P1_1 \wedge P1_2$, (B) is the result of $P1_1 \vee P1_3$, and (C) is the result of NOT $P1_3$ (or $P1_3'$), where $P1_1$, $P1_2$, and $P1_3$ are shown in Fig. 2.

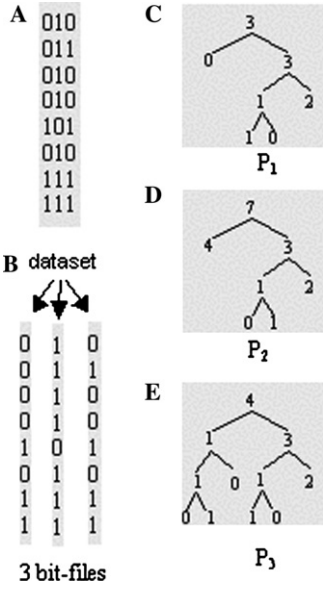

Fig. 1. Construction of 1-D basic P-trees.

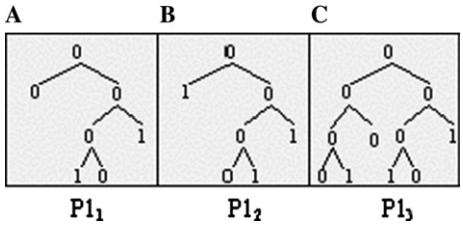

Fig. 2. Results of predicate trees.

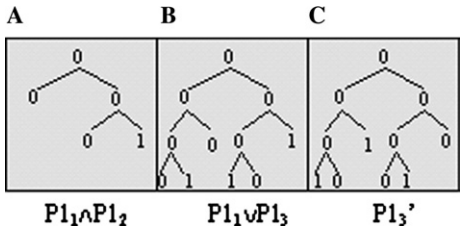

Fig. 3. AND, OR, and NOT operations.

3.2. Optimized range P-tree operations

In this section, we present several original propositions for optimized range predicate operations using basic predicate trees to calculate the nearest neighbors. Range predicate tree, $P_{x \prec y}$, is a basic predicate tree that satisfies predicate $x \prec y$, where y is a boundary value, and $\prec$ is a comparison operator, i.e., $<$, $>$, $\geqslant$, and $\leqslant$. Without loss of generality, we only present the calculation of range predicate $P_{A>c}$, $P_{A\leqslant c}$, $P_{c1<A\leqslant c2}$ and their proof as follows.

Lemma 1. *Let* P_1, P_2 *be two basic predicate* P*-trees, and* P_1' *is the complement* P*-tree of* P_1 *by complementing each pure node of* P_1, *then* $P_1 \vee (P_1' \wedge P_2) = P_1 \vee P_2$ *and* $P_1 \wedge (P_1' \vee P_2) = P_1 \wedge P_2$.

Proof.

$P_1 \vee (P_1' \wedge P_2)$
(According to the distribution property of P-tree operations)
$= (P_1 \vee P_1') \wedge (P_1 \vee P_2)$
$=$ True $\wedge (P_1 \vee P_2)$
$= P_1 \vee P_2$
Similarly, $P_1 \wedge (P_1' \vee P_2) = P_1 \wedge P_2$.

Proposition 1. *Let A be jth attribute of data set X, m be its bit-width, and* $P_m, P_{m-1}, \ldots, P_0$ *be the basic* P*-trees for the vertical bit files of A. Let* $c = b_m \cdots b_i \cdots b_0$, *where* b_i *is ith binary bit value of c, and* $P_{A>c}$ *be the predicate tree for the predicate* $A > c$, *then*

$$P_{A>c} = P_m \mathrm{op}_m \cdots P_i \mathrm{op}_i P_{i-1} \cdots \mathrm{op}_{k+1} P_k, \quad k \leqslant i \leqslant m,$$

where (1) op_i *is* $\wedge$ *if* $b_i = 1$, op_i *is* $\vee$ *otherwise*, (2) k *is the rightmost bit position with value of* "0," *i.e.*, $b_k = 0$,

$b_j = 1$, $\forall j < k$, *and* (3) *the operators are right binding. Here the right binding means operators are associated from right to left, e.g.,* P_2 op2 P_1 op1 P_0 *is equivalent to* (P_2 op2 (P_1 op1 P_0)).

Proof (*By induction on number of bits*).

Base case: without loss of generality, assume $b_1 = 1$, then need show $P_{A>c} = P_2$ op2 P_1 holds. If $b_2 = 1$, obviously the predicate tree for $A > (11)_2$ is $P_{A>c} = P_1 \wedge P_0$. If $b_2 = 0$, the predicate tree for $A > (01)_2$ is $P_{A>c} = P_2 \vee (P'_2 \wedge P_1)$. According to Lemma 1, we get $P_{A>c} = P_2 \vee P_1$ holds.

Inductive step: assume $P_{A>c} = P_n\, op_n \cdots P_k$, we need to show $P_{A>c} = P_{n+1} op_{n+1} P_n\, op_n \cdots P_k$ holds. Let $P_{\text{right}} = P_n\, op_n \cdots P_k$, if $b_{n+1} = 1$, then obviously $P_{A>c} = P_{n+1} \wedge P_{\text{right}}$. If $b_{n+1} = 0$, then $P_{A>c} = P_{n+1} \vee (P'_{n+1} \wedge P_{\text{right}})$. According to Lemma 1, we get $P_{A>c} = P_{n+1} \vee P_{\text{right}}$ holds.

Proposition 2. *Let A be jth attribute of data set X, m be its bit-width, and* $P_m, P_{m-1}, \ldots, P_0$ *be the basic* P*-trees for the vertical bit files of A. Let* $c = b_m \cdots b_i \cdots b_0$, *where* b_i *is ith binary bit value of c, and* $P_{A \leqslant c}$ *be the predicate tree for* $A \leqslant c$, *then*

$$P_{A \leqslant c} = P'_m op_m, \ldots, P'_i op_i P'_{i-1}, \ldots, op_{k+1} P'_k, \quad k \leqslant i \leqslant m,$$

where (1) op_i *is* $\wedge$ *if* $b_i = 0$, op_i *is* $\vee$ *otherwise*, (2) *k is the rightmost bit position with value of* "0," *i.e.,* $b_k = 0$, $b_j = 1$, $\forall j < k$, *and* (3) *the operators are right binding.*

Proof (*By induction on number of bits*).

Base case: without loss of generality, assume $b_0 = 0$, then need show $P_{A \leqslant c} = P'_1 op1 P'_0$ holds. If $b_1 = 0$, obviously the predicate tree for $A \leqslant (00)_2$ is $P_{A \leqslant c} = P'_1 \wedge P'_0$. If $b_1 = 1$, the predicate tree for $A \leqslant (10)_2$ is $P_{A \leqslant c} = P'_1 \vee (P_1 \wedge P'_0)$. According to Lemma 1, we get $P_{A \leqslant c} = P'_1 \vee P'_0$ holds.

Inductive step: assume $P_{A \leqslant c} = P'_n op_n \cdots P'_k$, we need to show $P_{A \leqslant c} = P'_{n+1} op_{n+1} P'_n op_n \cdots P'_k$ holds. Let $P_{\text{right}} = P'_n op_n, \ldots, P'_k$, if $b_{n+1} = 0$, then obviously $P_{A \leqslant c} = P'_{n+1} \wedge P_{\text{right}}$. If $b_{n+1} = 1$, then $P_{A \leqslant c} = P'_{n+1} \vee (P_{n+1} \wedge P_{\text{right}})$. According to Lemma 1, we get $P_{A \leqslant c} = P'_{n+1} \vee P_{\text{right}}$ holds.

Proposition 3. *Let A be jth attribute of data set X,* $P_{A \leqslant c}$ *and* $P_{A>c}$ *are the predicate tree for* $A \leqslant c$ *and* $A > c$, *where c is a boundary value, then* $P_{A \leqslant c} = P'_{A>c}$.

Proof. Obvious true by checking the Proposition 1 and 2 according to $\wedge = \vee'$ and $P_m = (P'_m)'$.

Proposition 4. *Given the same assumption of A and its* P*-trees. Suppose* $m - r + 1$ *high order bits of bound value c*1 *and c*2 *are the same, then we have* $c1 = b_m \cdots b_r b1_{r-1} \cdots b1_1$, $c2 = b_m \cdots b_r b2_{r-1} \cdots b2_1$. *Let* $s1 = b1_{r-1}, \ldots, b1_1$, $s2 = b2_{r-1}, \ldots, b2_1$, *and B be the value of low* $r - 1$ *bits of A, then predicate interval tree,* $P_{c1<A \leqslant c2}$, *is calculated as*

$$P_{c1<A \leqslant c2} = \eta_m \wedge \eta_{m-1} \wedge \cdots \eta_r \wedge P_{B>s1} \wedge P_{B \leqslant s2},$$

where η_i *is* P_i *if* $b_i = 1$, η_i *is* P'_i *otherwise.* $P_{B>s1}$ *and* $P_{B \leqslant s2}$ *are calculated according to Propositions* 1 *and* 2, *respectively.*

Proof. According to Propositions 1 and 2, we have $P_{A>c1} = P_m op1_m \cdots P_r op1_r\ P1_{r-1} \cdots op1_{k+1}\ P1_k$, $P_{A \leqslant c2} = P'_m op2_m, \ldots, P'_r op2_r P2'_{r-1}, \ldots, op2_{k+1} P2'_k$, where $op1_i$ is if $b1_i = 1$ and $op2_i$ is $\vee$ if $b2_i = 1$, $op1_i$ is $\vee$ and $op2_i$ is $\wedge$ otherwise. We observe that if $b1_i = b2_i$, $op1_i$ and $op2_i$ are opposite. This is where we can further optimize. Suppose $b_m = 1$, then $op1_m$ is $\wedge$, $op2_m$ is $\vee$, hence

$$\begin{aligned} P_{c1<A \leqslant c2} &= P_{A>c1} \wedge P_{A \leqslant c2} \\ &= (P_m \wedge \cdots P_r op1_r P1_{r-1} \cdots op1_{k+1} P1_k) \\ &\quad \wedge (P'_m \vee \cdots P'_r op2_r P2'_{r-1} \cdots op2_{k+1} P2'_k) \\ &= \langle \text{associative properties of } \wedge \text{ and } \vee \rangle \end{aligned}$$

$$\begin{aligned} &P_m \wedge (P'_m \vee P'_{m-1} \cdots P'_r op2_r P2'_{r-1} \cdots op2_{k+1} P2'_k) \\ &\quad \wedge (P_{m-1} op1_{m-1} \cdots P_r op1_r P1_{r-1} \cdots op1_{k+1} P1_k) \\ &= \langle \text{Apply Lemma}(m-r) \text{ th times} \rangle \end{aligned}$$

$$\begin{aligned} &P_m \wedge P_{m-1} \wedge \ldots P_r \wedge (P1_{r-1} op1_{r-1} \cdots op1_{k+1} P1_k) \\ &\quad \wedge (P2'_{r-1} op2_{r-1} \ldots op2_{k+1} P2'_k) \\ &= \langle \text{Proposition 1 and Proposition 2} \rangle \end{aligned}$$

$$P_m \wedge P_{m-1} \wedge \cdots P_r \wedge P_{B>s1} \wedge P_{B \leqslant s2}$$

Similarly, we can approve the case when $b_m = 0$

$$\begin{aligned} P_{c1<A \leqslant c2} &= P_{A>c1} \wedge P_{A \leqslant c2} \\ &= (P_m \vee \cdots P_r op1_r P1_{r-1} \cdots op1_{k+1} P1_k) \\ &\quad \wedge (P'_m \wedge \cdots P'_r op2_r P2'_{r-1} \cdots op2_{k+1} P2'_k) \\ &= \langle \text{Apply Lemma}(m-r) \text{ th times} \rangle \end{aligned}$$

$$\begin{aligned} &P'_m \wedge P'_{m-1} \wedge \cdots P'_r \wedge (P1_{r-1} op1_{r-1} \cdots op1_{k+1} P1_k) \\ &\quad \wedge (P2'_{r-1} op2_{r-1} \cdots op2_{k+1} P2'_k) \\ &= \langle \text{Proposition 1 and Proposition 2} \rangle \end{aligned}$$

$$P'_m \wedge P'_{m-1} \wedge \cdots P'_r \wedge P_{B>s1} \wedge P_{B \leqslant s2}$$

Hold.

4. The EIN-ring KNN/LSVM classification

One of the key advantages of nearest neighbor approaches is that it estimates the density function for each target instance object locally and differently instead of estimating once for the entire instance space [10]. In this section, we present a comprehensive vertical sampled-based KNN/LSVM classification approach. We first develop an efficient neighborhood search technique using equal interval neighborhood ring (EIN-ring) in Section 4.1, and then propose an efficient weighted EIN-ring

KNN classification approach and exploit it as the fit function for genetic algorithms to select the most related subset of features in Section 4.2. Finally, we propose a local boundary-based classification approach, local support vector machine to improve the classification accuracy in Section 4.3.

4.1. EIN-ring neighborhood search

We exploit nearest neighborhood classifier as the fit function of genetic algorithms to select the most related subset of genes for high-dimensional gene expression data. Due to the extensive computational requirement of genetic algorithms, it is crucial to improve the efficiency of the evaluation of fit function, which is the major computational cost of genetic algorithms. In this section, we develop a novel efficient neighborhood search technique using P-trees, called EIN-ring approach.

Definition 1. The neighborhood ring of data instance c with radii r_1 and r_2 is defined as the set $R(c,r_1,r_2) = \{x \in X | r_1 < |c-x| \leqslant r_2\}$, where $|c-x|$ is the distance between x and c.

Definition 2. The equal interval neighborhood ring of data instance c with radii r and fixed interval λ is defined as the neighborhood ring $R(c,r,r+\lambda) = \{x \in X | r < |c-x| \leqslant r+\lambda\}$, where $|c-x|$ is the distance between x and c. For $r = k\lambda$, $k = 1, 2, \ldots$, the ring is called the kth EIN-ring. Fig. 4 shows 2-d EIN-rings with $k = 1, 2$, and 3.

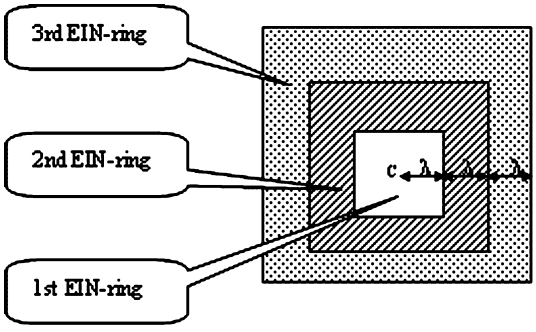

Fig. 4. Diagram of EIN-rings.

The interval λ could be a fixed equal interval or geometric interval with a fixed factor. Note that the geometric interval with a factor of two turns out to be a special metric, called HOBbit metric [18], which can be calculated extremely fast using P-trees. The interval can be adaptively adjusted with respect to the sparseness of the dataset. The calculation of neighbors within EIN-ring $R(c,r,r+1)$ is as follows.

Let $P_{r,\lambda}$ be the P-tree representing data instances within EIN-ring $R(c,r,r+\lambda)$. Note that $P_{r,\lambda}$ is just the predicate tree corresponding to the predicate $c-r-\lambda < X \leqslant c-r$ or $c+r < X \leqslant c+r+\lambda$. We first calculate the data instances within neighborhood ring $R(c,0,r)$ and $R(c,0,r+\lambda)$ by $P_{c-r<X\leqslant c+r}$ and $P'_{c-r-\lambda<X\leqslant x+c+r+\lambda}$, respectively. $P_{c-r-\lambda<X\leqslant c+r+\lambda}$ is shown as the shadow area of (A), and $P'_{c-r<X\leqslant c+r}$ is the shadow area of (B) in Fig. 5. The data instances within the EIN-ring $R(c,r,r+\lambda)$ are those in $R(c,0,r+\lambda)$ but not in $R(c,0,r)$. Therefore, $P_{r,\lambda}$ is calculated by the following formula:

$$P_{r,\lambda} = P_{c-r-\lambda<X\leqslant c+r+\lambda} \wedge P'_{c-r<X\leqslant c+r}, \quad (1)$$

4.2. Weighted EIN-ring KNN classification approach

The KNN classification approach is based on the assumption that an unclassified target instance is similar to the instances that are nearby in the feature space. The weighted EIN-ring KNN classification has two major steps. First, the k nearest neighbors are selected from the training data instances within successive EIN-rings for an unclassified target instance. Second, the unclassified target instance is assigned to the winning class according to the weighted majority vote. The weights of feature dimensions are optimized by a genetic algorithm with EIN-ring KNN as the fitness function. The genetic algorithm is a classical global and nonlinear optimization method that is derided by analogy to evolution and natural genetics [20]. If the weight of one feature dimension is zero, it means that feature is not selected for classification vote, otherwise selected. The value of the weight indicates how important the corresponding feature is in the classification vote. The classification process of weighted EIN-ring KNN approach is illustrated in Fig. 6.

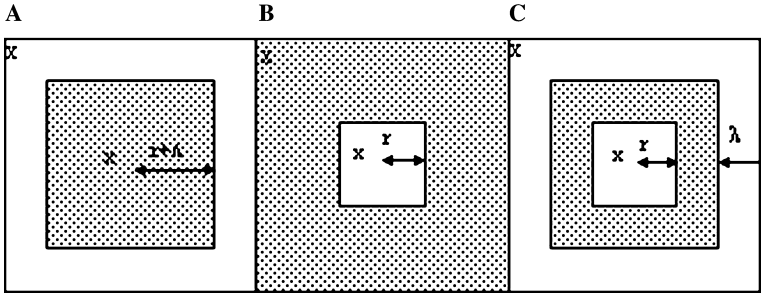

Fig. 5. Calculation of data points within EIN-ring $R(x,r,r+\lambda)$.

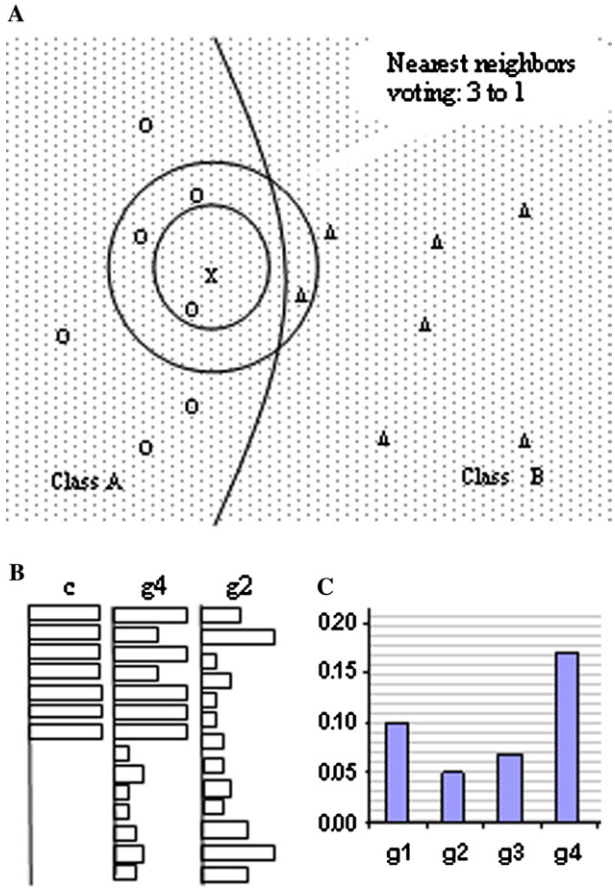

Fig. 6. Diagram of weighted EIN-ring KNN approach with $k = 3$.

In Fig. 6A, data instance x is the target instance to be classified between two classes, A and B, by means of weighted EIN-ring KNN with $k = 3$. The relative similarity among the data instances is calculated based on weighted distance in which different genes have different discriminant importance for classification. Panel B shows two extreme genes, the most relevant and the least relevant gene, and panel C shows weights assigned to genes. Within the first ring, we only got one nearest neighbor, less than three. Then we calculate the next successive neighborhood ring and get four neighbor instances. Since the total number of neighbors are greater than three, we then stop calculating neighborhood rings and check the voting score. Because of three instances of class A and one instance of class B, we then assign instance x to class A according to majority rule.

The parameter of k is selected by checking the vote score of each neighborhood ring following a simple "early" stop rule, i.e., stopping the neighborhood calculation as soon as the neighborhood ring becomes indecisiveness or dominating class within the neighborhood ring changes. A neighborhood ring is called decisive if one class instance within the ring dominates. Fig. 7 illustrates a decisive ring and an indecisive ring among the two-dimensional EIN-rings.

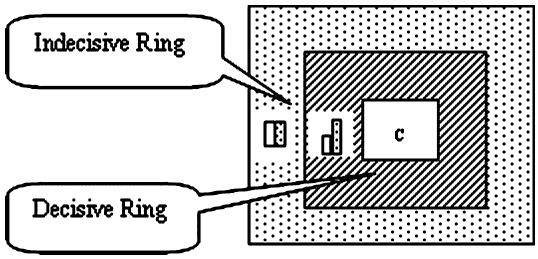

Fig. 7. Histogram within neighborhood rings.

The decisiveness within a neighborhood ring is measured by the vote score, which is calculated by the height of the winner bin minus the others and then divided by the sum of heights of all histogram bins. For a dataset with a number of C classes, we first create a mask P-tree for the ith class, PC_i, in which a "1" value indicates that the corresponding data instance has the ith class label and a "0" value indicates otherwise. The data instances with ith class label within kth EIN-ring, $R(x, k\lambda, (k+1)\lambda)$, is calculated as

$$\mathrm{PN}_{r,i} = \mathrm{P}_{r,\lambda} \wedge \mathrm{PC}_i, \tag{2}$$

where $r = k\lambda$. $\mathrm{P}_{r,\lambda}$ is calculated according to Eq. (1). Let rc() be the root count function that returns the number of ones in a P-tree, then the vote score within the kth EIN-ring, $R(x, k\lambda, (k+1)\lambda)$, is calculated using P-tree as

$$\mathrm{VS}_{r,i} = \frac{rc(\mathrm{PN}_{r,i})}{rc(\mathrm{P}_{r,\lambda})}. \tag{3}$$

The weights of feature dimensions are implemented by creating a weight string through which the P-trees that participate in the vote score calculation are selected. The weight string has a length of $d{*}\mathrm{m}_i$, where d is the number of dimensions and m_i is the coding string length for ith dimension. The "1" in the weight string means the corresponding P-tree will participate in vote score calculation, and "0" otherwise. By adjusting the number of P-trees that participate in voting using genetic algorithms, the weight string plays the role of weighting and selecting each feature dimension.

The process of optimizing weights using genetic algorithms is described as follows. Initially a population of weight strings is randomly generated, and classification error using weighted EIN-ring KNN is calculated through a k-fold crossvalidation, i.e., randomly dividing the dataset into k groups, taking turns to choose one group as the test data and the rest as training data, and calculating the average classification error. The weight strings that have small classification errors are chosen to reproduce by exchanging partial bits between each two weight strings with a small probability P_c and mutation, i.e., reverse some bits of a weight string randomly with probability P_m. The offspring weight strings are evaluated again, and reproduces the next generation until meeting stop condition, e.g., the maximum number

of generations or minimum improvement between consecutive generations. The best weight string with smallest classification error is selected as the final weight string.

One simple way of selection of the dimensions is to select the first d-dimension with largest weight. An alternative way is to transform the neighborhood data instances into d-dimensional space using Schoenberg method [20]. The advantage of the latter approach is that all the dimension information of the data are transformed into d-dimension through the weighted distance metric. Briefly, start with the distance matrix $D = [d_{ij}]$ (ith row and jth column of D) of the target data and its neighbors, and calculate the eigenvector of a symmetric metric B = HAH, where $A = [a_{ij}], a_{ij} = -d_{ij}^2/2$, and H is same size diagonal metric with $(1/k+1)$ on diagonal and $(1/k+1)$ off diagonal. The values in first dth eigenvectors with the largest eigenvalue is the new coordinate of target data and its corresponding neighbors.

4.3. On improving accuracy—LSVM approach

As mentioned earlier, our KNN/LSVM approach is motivated from the lesson of KDD Cup 2002. What we learned from task2 of KDD Cup 2002 is that KNN voting approach does not work well for narrow problem, so we developed a local proximal support vector machine (LSVM) to improve the classification accuracy. Instead of solving global classification boundary using nonlinear programming approach [21,22], the LSVM fits the classification boundary using piecewise segment hyperplanes based on local support vectors, as illustrated in Fig. 8.

In Fig. 8, there are two classes, A and B. The x is an unclassified instance, and S1, S2, S3, and S4 are the four nearest neighbors to the data instance x, which are used to form the local support vectors and to estimate the class boundary around the unclassified data instance x. The line through two data points M1 and M2 within the line segments S1S2 and S3S4 is the estimation of class boundary for the case of two dimensions.

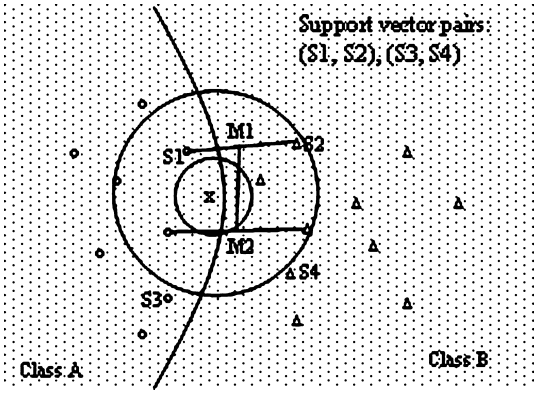

Fig. 8. Local support vectors approach.

The LSVM approach has two major steps. The first step is to find support vector pairs and to calculate the EIN-ring membership of them. The EIN-ring approach is used to find support vector pairs around the data instance x. The support vector pair is a pair of data instance that are mutual nearest neighbor with different class label. Specifically, a pair of data instance $x_i, x_j \in X$, $i \neq j$, is the support vector pair, denoted as $SVP(x_i, x_j)$, if and only if $d(x_i, x_j) \leqslant d(x_k, x_1)$ $x_k, x_1 \in X$ and $x_k \in c1$, $x_1 \in c2$.

Given the radius of the neighborhood, we define the EIN-ring memberships of a data x as the weighted summation of vote score $VS_{r,i}$, where the $VS_{r,j}$ is the ratio of the number of neighbors with the ith class label to the total number of neighbors. The EIN-ring memberships within neighborhood σ is calculated as follows:

$$M_{\sigma,i} = \sum_{r=1}^{\sigma} w_r * VS_{r,i}, \tag{4}$$

where $VS_{r,i}$ is calculated according to Eq. (3), σ is the radius of the neighborhood, and w_r is the weight of the kth EIN-ring, $w_r = 1 - k\frac{\lambda}{\sigma}$, $r = k\lambda$. There are many other kernel functions that can be used to weight the shape of locality, such as linear kernel, polynomial kernel, RBF-kernel.

The second step is to fit hyperplane and assign a class label to the target data instance. The hyperplane spans over the data points between each SVP. We define these data point as boundary sentry (BS). The boundary sentry between support vector pair (x_i, x_j), denoted as $BS_{i,j}$, is calculated as

$$BS_{i,j} = \lambda x_i + (1 - \lambda)x_j, \tag{5}$$

where $\lambda = M_{\sigma,j}/(M_{\sigma,i} + M_{\sigma,j})$, $M_{\sigma,i}$ and $M_{\sigma,j}$ are calculated according to Eq. (4). Given a test data instance x with d dimension, the boundary hyperplane is determined by d boundary sentries. For example, if $d = 2$, the boundary is a line and we need two boundary sentries. Similarly, if $d = 3$, we need three boundary sentries to determine the boundary plane.

After fitting the class boundary with piecewise hyperplane, we check if the data instance of support vectors have the same class on the same side of class boundary. If not, we replace the misleading support vector with the next nearest one and check the class boundary untill the data instance of support vectors have the same class on the same side of class boundary. Finally, we determine the class label of x based on its relative location to the boundary hyperplane.

5. Performance study

We compared our EIN-ring KNN/LSVM classification approach with Fisher's linear discriminant analysis (FLDA) [21] and Golub's weighted voting method [2]. The two test datasets we selected are Leukemia dataset

and Lymphoma dataset, denoted as DB1 and DB2, respectively, which were prepared in the same fashion as described in paper [6]. The accuracy is measured by precision, which is calculated for every class c as TP/(TP + FP), where TP (true positives) is the number of test instances that are correctly classified as c and FP (false positives) is the number of test instances that should be classified as c but not. The precision comparison with FLDA and Golub's weighted KNN approach is based on the precision report on DB1 in paper [6]. The efficiency is compared between our algorithm by P-tree data representation and using double precision numbers (DPN) in our algorithm.

We implemented KNN/LSVM approach in the C language and run on a 1 GHz Pentium PC machine with 1GB main memory, and Debian Linux 4.0. In this experiment, we chose uniform crossover, stochastic universal sampling selection, leave-one-out crossvalidation, $P_c = 0.5$, $P_m = 0.05$, $k = 5$ and GA population size of 200, strength length of 8 based on preliminary experiment. Termination condition is always checked after selection, mutation, and re-evaluation, which is set to 1000 maximum runs and minimum difference of fitness value $E = 0.01$ between two generations.

The precision comparison with FLDA and Golub's KNN on DB1 is plotted in the upper panel of Fig. 9. The overall run time of KNN/LSVM and the same approach using DPN is shown in the lower panel of Fig. 9. In general, the nearest neighbor and KNN/LSVM had the higher precision strength than FLDA. The possible reason for the poor performance of FLDA is that it is a "global" approach that is not well suited to high-dimensional skew data, while nearest neighbor and our approach methods make optimal decision at local level. Compared to the precision of FLDA and Golub's KNN approach on DB2, KNN/LSVM achieved relatively higher precision than FLDA and comparable precision to Golub's KNN. As for efficiency, it is clear that P-tree-based KNN/LSVM is significant faster than the same approach without using P-trees. Drastic speed improvement of KNN with P-trees is observed when the size of the data is very large, as shown in the case of $5000 \times 20{,}000$ matrix size in [18].

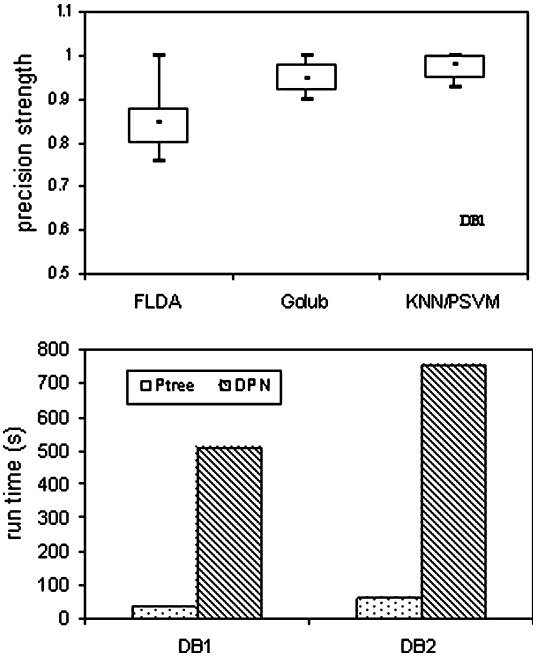

Fig. 9. Comparison of accuracy and run time.

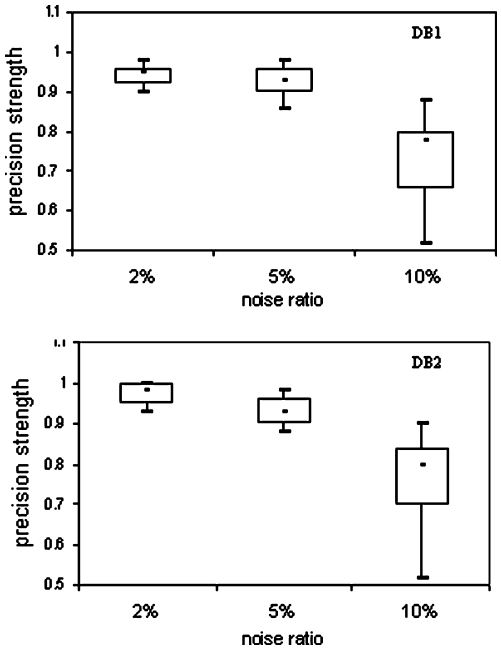

Fig. 10. Precision strength measurements on DB1 and DB2 with 2, 5, and 10% noises.

We tested the sensitivity of our algorithm under various noise environments by adding 2, 5, and 10% uniform random noise to DB1 and DB2. The comparison of precision measurements of KNN/LSVM under different noise is plotted in Fig. 10. Comparing to the case without noise, the average precision measurement of KNN/LSVM under 2 and 5% noise change slightly, while the average precision measurement of KNN/LSVM approach under 10% noises decreases dramatically. The range of precision measurement spreads slightly under 2 and 5% noises, and more widely under 10% noises. The result indicates that KNN/LSVM is robust under small and moderate noise, which inherits from KNN voting scheme and the LSVM. The robustness capability of KNN/LSVM to high-dimensional noises is a highly desirable characteristic for gene expression data analysis.

6. Conclusion

In this paper, we have proposed a comprehensive vertical sample-based classification approach, KNN/

LSVM, characterized by P-tree, combination of majority voting and boundary approach, and weights optimization using the genetic algorithm. Experiments with public microarray data demonstrated that our approach can achieve high accuracy and efficiency, hence could be a powerful tool for gene expression data analysis.

In addition to improved performance, our approach also showed strong robustness to noises in high-dimensional data. The reason for that is mainly related to the property of KNN majority voting scheme, which is highly desirable for gene expression analysis.

In the future, we will apply this approach to large-scale time series gene expression data, where the efficient and scalable analysis approach is in demand. We will also investigate the influence of the partition on the balance of accuracy and computation efficiency.

References

[1] Eisen MB, Spellman PT. Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci USA 1995:14863–8.
[2] Golub TR, Slonim DK, Tamayo P, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 1999;286:531–7.
[3] Li L, Weinberg CR, Darden TA, Pedersen LG. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. Bioinformatics 2001;17:1131–42.
[4] Furey TS. Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics 2000;16(10):906–14.
[5] Yang YH, Dudoit S, Luu P, et al. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. Nucleic Acids Res 2002;30(4):e15.
[6] Dudoit S, Fridyland JF, Speed TP. Comparison of discrimination methods for tumor classification based on microarray data. J Am Stat Assoc 2002:77–87.
[7] Perera A, Denton A, Kotala P, Jockheck W, Granda WV, Perrizo W. P-tree classification of yeast gene deletion data. SIGKDD Explor 2003;4(2):108–9.
[8] Brown PS, Grundy WN. Support vector machine classification of microarray gene expression data. Proc Natl Acad Sci USA 2000;97:262–7.
[9] Aha D, Kibler D, Albert M. Instance-based learning algorithms. Mach Learn 1991;6(1):37–66.
[10] T. Mitchell, Machine Learning. Morgan Kaufmann; 1997.
[11] Cover TM, Hart PE. Nearest neighbor pattern classification. Trans IEEE Inform Theory 1967;IT-13:21–7.
[12] T.M. Cover, Rates of convergence for nearest neighbor procedures, in: Proceedings of the Hawaii International Conference on System Sciences, 1968, p. 413–5.
[13] Wagner TJ. Convergence of the nearest neighbor rule. Trans IEEE Inform Theory 1971;IT-17:566–71.
[14] Moore DS, Yackel JW. Consistency properties of nearest neighbor density estimates. Ann Stat 1977;5:143–54.
[15] Devroye L, Wagner TJ. The strong uniform consistency of nearest neighbor density estimates. Ann Stat 1977;5:536–40.
[16] Jain AK, Duin RPW, Mao J. Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 2000;22(1):4–37.
[17] Ooi CH, Tan P. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. Bioinformatics 2003;19:37–44.
[18] Ding Q, Khan M, Roy A, Perrizo W. The P-tree algebra. ACM Symp Appl Comput 2002:11–4.
[19] Perrizo W. Peano Count Tree Technology. Technical Report NDSU-CSOR-TR-01-1; 2001.
[20] Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins G, editor. Foundations of genetic algorithms. Berlin: Morgan Kaufmann; 1991. p. 69–93.
[21] Vapnik V. The nature of statistical learning theory. New York: Springer-Verlag; 1995.
[22] Mangasarian OL, Musicant DR. Data discrimination via nonlinear generalized support vector machines. Technical Report 99–03, Computer Sciences Department, University of Wisconsin, 1999.