Microarray Data Classifier Consisting of *k*-Top-Scoring Rank-Comparison Decision Rules With a Variable Number of Genes

Youngmi Yoon, Sangjay Bien, and Sanghyun Park

***Abstract*—Microarray experiments generate quantitative expression measurements for thousands of genes simultaneously, which is useful for phenotype classification of many diseases. Our proposed phenotype classifier is an ensemble method with *k*-top-scoring decision rules. Each rule involves a number of genes, a rank comparison relation among them, and a class label. Current classifiers, which are also ensemble methods, consist of *k*-top-scoring decision rules. Some of these classifiers fix the number of genes in each rule as a triple or a pair. In this paper, we generalize the number of genes involved in each rule. The number of genes in each rule ranges from 2 to *N*, respectively. Generalizing the number of genes increases the robustness and the reliability of the classifier for the class prediction of an independent sample. Our algorithm saves resources by combining shorter rules in order to build a longer rule. It converges rapidly toward its high-scoring rule list by implementing several heuristics. The parameter *k* is determined by applying leave-one-out cross validation to the training dataset.**

***Index Terms*—Data mining, knowledge-based data mining, microarray data analysis, microarray data classification.**

I. Introduction

DNA microarray technology makes it possible to simultaneously examine tens of thousands of gene expression values in a single experiment. A DNA microarray is a collection of microscopic DNA spots, commonly representing single genes, which are arrayed on a solid surface. Quantitative measurements of DNA arrays are obtained after many processing and normalization steps. Gene expression levels can be determined for samples taken: 1) at multiple time instants of a biological process (e.g., different phases of cell division) or 2) under various conditions (e.g., tumor samples with different histopathological diagnosis) [1]. Microarray data, which we are focused on, takes the form of Fig. 1. Each row represents a gene, each column represents a sample, and each cell holds the quantitative expression value of a sample in the gene. A sample has its own class label, normal or tumor, and a set of expression values corresponding to each gene. A microarray dataset consists of one set of samples labeled as normal and another set labeled as tumor.

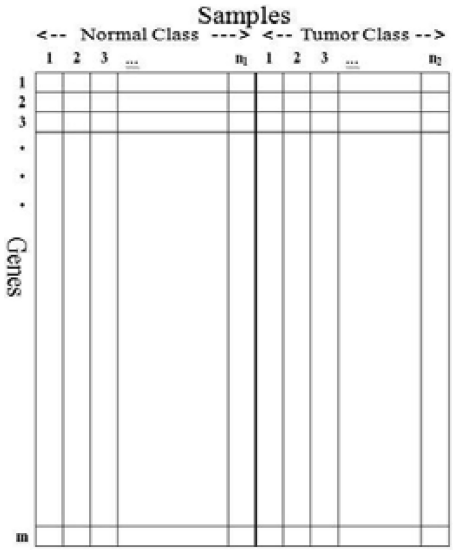

Fig. 1. Microarray dataset.

Classification is the process of finding a model for class attribute as a function of the gene expression values given a collection of samples as a training dataset [2]. Microarrays are useful tools for the phenotype classification of many cancer-related diseases. A limitation of microarray classification is that there are not enough samples per experimental study provided in order to derive statistically significant results [3], [4].

The number of relevant genes that exert an influence on classification is restricted. Most of the genes are unrelated to the given phenotype classification problem [5]. The key informative features (genes) represent a base of reduced cardinality for subsequent analysis such as phenotype classification [6]. The minimum number of genes required to build a robust classifier is generally between 20 and 70.

The algorithm for integrating multiple microarray datasets holding same biological objectives and selecting informative genes is from the previous work of Yoon *et al.* [7]. They converted the expression value of a gene to a rank value within a sample. This is an effective method of enlarging the number of samples. She identified informative genes by calculating the number of swaps to reach a perfectly split sequence for each gene. This approach mitigates the *curse of dimensionality*

Manuscript received March 18, 2008; revised October 5, 2008 and June 1, 2009 and October 28, 2009. First published December 22, 2009; current version published February 18, 2010. This work was supported by the National Research Foundation of Korea funded by the Ministry of Science and Technology, Korea Government under Grant 2009-0083311. This paper was recommended by Associate Editor Y. Jin.

Y. Yoon is with the Department of Information Technology, Gachon University of Medicine and Science, Incheon 406-799, Korea (e-mail: ymyoon@gachon.ac.kr).

S. Bien is with the Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul 151-742, Korea (e-mail: sayaya@snu.ac.kr).

S. Park is with the Department of Computer Science, Yonsei University, Seoul 120-749, Korea (e-mail: sanghyun@cs.yonsei.ac.kr).

Digital Object Identifier 10.1109/TSMCC.2009.2036594

problem by enlarging the number of samples via integration and by reducing the number of genes via filtering of informative genes. The gene selection approaches based on probability of selection [8], entropy [9], and gradient-leave-one-out gene selection (GLGS) algorithm [10] can provide gene subsets, leading to more accurate classification results.

Microarray datasets produced by two different laboratories that have the same biological objectives, even when the same platform is used, are affected by numerous sources of nonbiological variation and might retain "lab effects." This rank-based integration method offers a natural way to overcome the heterogeneity among multiple datasets, and therefore, to extract, compare, and integrate their information. However, it should be noted that some information could be lost in the process of rank transformation [11]. Since the method transforms the actual expression values into ranks within a sample, it can integrate datasets produced by a wide variety of platforms, such as Affymetrix oligonucleotide arrays, cDNA arrays, and other custom-made arrays. It has the ability to handle variability among datasets and can generate a single significance measurement for each gene.

This scheme only uses ranks rather than actual expression levels, and thus, there may be some slight information loss. However, the advantage is a gain in robustness to outliers, normalization schemes, and systematic errors such as chip-to-chip variation, and this provides scientists with a powerful tool to utilize the already existing microarray data resource.

The current classification methods from statistical learning are neural networks [12], [13], k-nearest neighbor (k-NN) [14], and support vector machine (SVM) [15]–[19]. Typically, multilayered neural networks result in predictions that are based on nonlinear functions of many expression values, and consequently, there are highly complex decision boundaries between the classes of interest. Such boundaries are difficult to summarize in simple terms, or to characterize in a manner that is biologically meaningful.

Although, decision tree [20], [21] uses a relatively small number of genes for classification, and this is not difficult to interpret in simple or biologically meaningful terms, this is known to be a sensitive type of classifier. Small perturbations in the training samples lead to large differences in its tree structure [22], [23]. Random forest is an algorithm for classification developed by Breiman [24] that uses an ensemble of classification trees, which is not as sensible as decision tree. The concept of ensemble learning is that the integration of multiple (unstable) classifiers will enhance the performance of the final classifier. Hence, an ensemble classifier can have better overall performance than the individual base classifiers [25], [26].

The existing methods are k-top-scoring triple (k-TST) [7], which is the previous work of Yoon *et al.*, and k-TS pair (k-TSP) [27] for a comparison-based ensemble machine learning approach. These are biologically intuitive and simple to interpret. In k-TST, a family of the k-top-scoring gene triples is isolated and the class of an independent sample is predicted based on the k-decision rules. This involves comparing the rank values in each gene triple, and then, aggregating the results. k-TSP also consists of k-decision rules that involve comparing the gene expression values in each pair. These current methods fix the number of genes involved in a rule as three or two.

TABLE I
NUMBER OF RULES AND EXPECTED TIME COMPLEXITY

	k-TSP	k-TST	k-TSN
Number of Possible Rules of Gene Combination	${}_nP_2$	${}_nP_3$	${}_nP_2 + {}_nP_3 + \ldots + {}_nP_n$
Time Complexity	$O(n^2)$	$O(n^3)$	$O(n^n)$

In this paper, we are proposing the k-top-scoring N (k-TSN) model that generalizes the number of genes involved in each rule. Generalizing the number of genes increases the robustness and reliability of the classifier for class prediction of an independent sample. Our classifier consists of k-decision rules. Each rule involves a number of genes, a rank comparison relation among them, and a class label. The number of genes involved in each rule ranges from 2 to N.

Unlike the current methods, the proposed method does not generate all of the possible rules of gene combination. We build a new longer rule by combining shorter rules while estimating the score range of the new rule. We stop consideration of the new rule if its estimated maximum score is below the minimum score of the current rule list. This saves computing time and memory space. We implement several heuristics to eliminate unnecessary rule combination. This ensures that a large amount of rules converge quickly to a k-high-scoring rule list and enables us to achieve a high classification accuracy.

To this end, the features of the k-TSN model for phenotype classification are explicitly presented throughout this paper. The following sections present the k-TSN model, algorithms, experimental results, and conclusion.

II. k-TSN MODEL

The k-TSN model, we are proposing, builds k-decision rules using training datasets and predicts the class of an independent sample as *tumor* or *normal*. In this section, we explain the motivation for our study, decision rules, score measurements concerning decision rule selection, and the k-TSN classifier.

A. Motivation

The basic idea of the k-TSN model is to generalize the number of genes involved in a decision rule, unlike current methods, such as k-TSP and k-TST. The current methods generate all the possible rules of gene combination, calculate the score for each rule, and select the top k-scoring rules. Direct application of the algorithm to our model is not appropriate. When the number of genes is n, Table I shows the expected time complexity if the same algorithm were applied to k-TSN. Because the run time of the algorithm is proportional to the number of rules generated for score calculation, the time complexity of k-TSN would be $O(n^n)$, which is not a feasible run time.

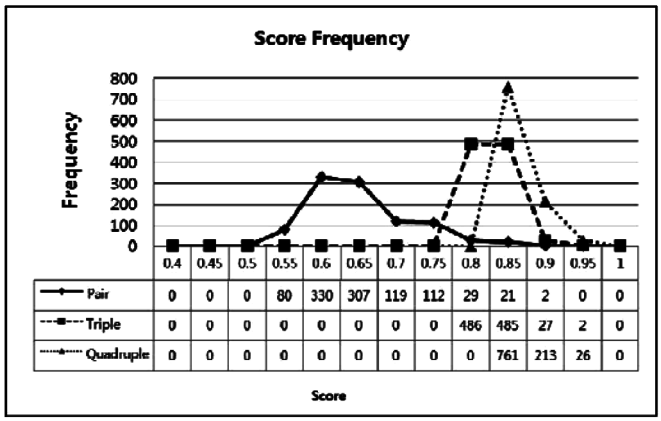

Fig. 2. Score frequency according to the number of genes in a rule.

Starting from two-gene rules, our approach builds longer rules while estimating the score of the longer rule and eliminating unnecessary rule generation. This can save a considerable amount of computation time.

Fig. 2 compares the score frequency of the pair rule, triple rule, and quadruple rule. The score represents the difference between the two probabilities of a rule's occurrence in one set of samples labeled as *normal* and another set labeled as *tumor*. A rule with a higher difference score is a more discriminative classification rule. We use Latulippe [28] as a training dataset, set the number of genes for a rule as pair, triple, and quadruple, select the 1000 top-scoring rules for each, and measure the score frequency for the three cases.

Fig. 2 shows the different score distributions, according to the number of genes involved in each rule. As the gene number increase from pair to quadruple, the latter has more high-scoring rules. High-scoring rules can better differentiate the two classes. Generalizing the number of genes increases the classification accuracy.

B. Rule

The expression value of a gene is given as a real number. This value can vary depending on the scale used by the individual biological labs. The expression value itself is not as important as the relative quantity that is closely connected with the class label of a sample. The rule in this paper is defined as: 1) rank comparison relation among involved genes and 2) class label. The number of genes involved in each rule ranges from 2 to *N*.

$$\text{Rule} = \{\text{Relation: } G_1 > G_2 > G_3 > \cdots > G_n;\ \text{Class: Tumor}\}.$$

C. Score

The score is calculated for every rule. The best set of rules is created using the training dataset. The score is defined as follows:

$$S_N = \mathit{NormalHit}/\mathit{NormalCount}$$
$$S_T = \mathit{TumorHit}/\mathit{TumorCount}$$

TABLE II
k-RULE LIST

No.	Rule	Score
1	$\{G_1 < G_2 < G_3;$ Normal$\}$	0.9
2	$\{G_6 < G_3;$ Tumor$\}$	0.87
:	:	:
k	$\{G_4 < G_1 < G_7;$ Normal$\}$	0.76

$$\text{Score} = \left\{ \begin{array}{l} S_N - S_T \ (\text{class label is } \textit{normal}) \\ S_T - S_N \ (\text{class label is } \textit{tumor}) \end{array} \right\}$$

where *NormalHit* (*TumorHit*) is the number of normal (tumor) samples that satisfies the relation of the rule among normal (tumor) samples in a training dataset, and *NormalCount* (*TumorCount*) is the total number of normal (tumor) samples in a training dataset.

The score ranges from 0 to 1. S_N denotes the *normal score* and S_T denotes *tumor score*. The difference between S_N and S_T is a score for each rule. High-scoring rules can better differentiate the two classes. If S_N is greater than S_T for a given gene relation, then the rule's class label is *normal*. Otherwise, it is *tumor*. Top-scoring rules are selected and these comprise the classifier.

D. k-TSN Classifier

Finally, a classifier consisting of the *k*-top-scoring decision rules is built, as shown in Table II. Each training dataset has its own optimal *k*-value, which is determined by leave-one-out cross validation (LOOCV) using 1% of informative genes. In order to avoid the optimistic bias that occurs when gene selection is done using the entire dataset rather than being done separately for each resampled training set, we selected 1% of newly informative genes for each step of LOOCV run and for construction of decision rules.

As *k*-decision rules are applied to a test sample, weighted majority voting is performed to predict the final class of the sample.

For performance evaluation of the algorithm, the predicted class is compared with the real class of the sample in the test dataset. The comparison is repeated for all of the samples in the test dataset, and the accuracy is calculated as follows:

$$\text{Accuracy} = \frac{\text{The number of correctly predicted samples}}{\text{The number of total samples}}.$$

III. ALGORITHM

Generating all possible rules of gene combinations from the training dataset and calculating the scores for all the rules requires a large amount of computation and resources. Building a new longer rule by combining shorter rules while estimating the score range of the new rule and stopping consideration of the new rule if its estimated maximum score is below the minimum score of the current rule list, can drastically save computing time and memory space. First, we generate all of the possible two-gene rules using 1% informative genes, which are the elements

TABLE III
TWO ELEMENT RULES

Rule	A	B
Relation	$G_1 < G_2$	$G_2 < G_3$
Class	Normal	Normal
S_N	*p*	*q*
S_T	*r*	*s*
Score	*p-r*	*q-s*

TABLE IV
NEW RULE BEING COMBINED

Rule	C	
Relation	$G_1<G_2<G_3$	
Class	Normal	
S_N (estimated)	Min	Max
	Max(p+q-1, 0)	*Min(p, q)*
S_T (estimated)	Min	Max
	Max(r+s-1, 0)	*Min(r, s)*
Score (estimated)	S_N - S_T	

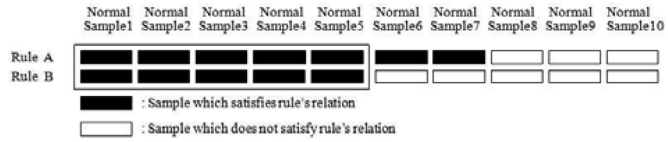

Fig. 3. Estimated maximum S_N score in rule C.

of further combination. Using these two-gene rules, we build longer gene rules with sizes ranging from 3 to *N*. Among the two-gene rules, the *k*-top-scoring rules are selected and inserted into the initial *k*-rule list. This section presents the details of the rule combination process, heuristics for rapid convergence toward a high-scoring rule list, and the algorithm summary.

A. Combination Process

Tables III and IV describe the rule combination process. Rules A and B are two element rules for which the class labels are all *normal*. We combine rules with the same class label. If we combine two rules with different class labels, the estimated score will be low; this is explained in Section III-C. If we consider the combination of two normal rules, S_N is the primary score and S_T is the secondary score. Let us assume that we build a new and longer rule C by combining rules A and B.

The new and longer rule C has the estimated values of S_N and S_T , as shown in Table IV.

First, we estimate the ranges of S_N and S_T, and then, we estimate the score for rule C. Rule C must satisfy $G_1 < G_2$ and $G_2 < G_3$. Since the S_N score of relation $G_1 < G_2$ is *p*, and the S_N score of relation $G_2 < G_3$ is *q*, the normal score of relation $G_1 < G_2 < G_3$ that satisfies both relations cannot exceed *p* or *q*. The maximum of S_N score is illustrated in Fig. 3.

Here, we estimate the minimum score of S_T of the longer rule. Even when the samples satisfying rule A overlap, as rarely as possible, with the samples satisfying rule B, there could be two samples overlapping each other, as shown in Fig. 4. The S_T score of rule A is *r*, which is 0.7, and the S_T score of rule B is

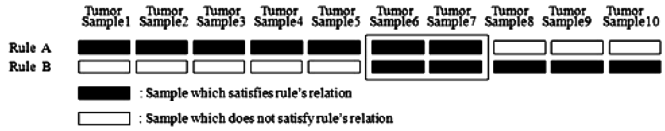

Fig. 4. Estimated minimum S_T score in rule C.

s, which is 0.5. The estimated minimum S_T score of rule C is 0.2, which is $(0.7 + 0.5 - 1)$. The ratio of the samples satisfying both rules can be obtained by adding each of the ratios and subtracting one, when the result is greater than or equal to one. If there is no overlap between the samples satisfying rule A and the samples satisfying rule B, then the estimated minimum score of S_T is zero. The S_T score of rule C is between 0 and 1 since it is also defined as *TumorHit/TumorCount*. For the sake of simplicity, this is denoted as $\text{Max}(r + s - 1, 0)$.

Accordingly, we can estimate the maximum score of rule C as follows:

$$\text{Score}_{\max} = \text{Min}(p, q) - \text{Max}(r + s - 1, 0).$$

If this estimated maximum score is greater than the minimum score of the *k*-rule list, then the real score of rule C is calculated. If the real score of rule C is greater than the minimum score of the *k*-rule list, then the *k*-rule list is updated with rule C, and the minimum score is also updated.

Subsequently, rule C can be used as an element for another high-scoring rule. Table IV implies that when rule C is to be combined with other rules as an element rule, the estimated maximum primary score of the new rule is *min*(*primary score of rule C, primary score of the other element rule*). Accordingly, if the calculated primary score of rule C is greater than the minimum score of the *k*-rule list, then rule C is inserted into the element rule list. The process of estimation and actual calculation has been done with the rule having the highest primary score and every other rule in the element rule list in each iteration. And then, the rules that have lower primary scores than the current minimum score are removed from the element rule list, and a new iteration starts.

As combination proceeds, the scores in the *k*-rule list converge to higher scores, while the number of rules in the element rule list is decreasing drastically, because in each iteration, the rules having lower primary scores than the current minimum score are removed from the element list.

The process of combining and updating the *k*-rule list and the element list continues until there are no more elements in the element rule list. At this stage, a classifier of the *k*-top-scoring decision rules has been built and the process of class prediction can start. Section III-D presents the details of the algorithm.

B. Class Prediction of Independent Samples

Each sample in the independent test dataset is checked to determine whether it satisfies each rule in the *k*-top-scoring decision rules built from the training dataset.

Majority voting is an approach that combines the predictions of multiple rules and obtains the final prediction. In the case

TABLE V
WEIGHTED MAJORITY VOTING FOR CLASS PREDICTION OF A SAMPLE WHEN $k = 5$

Rule	Score	Class Label	Relation Satisfied?	Cumulative Normal Score	Cumulative Tumor Score	Final Prediction
1	1.00	Tumor	Yes	0	1	
2	0.98	Normal	Yes	0.98	1	
3	0.95	Normal	No	0.98	1.95	Tumor
4	0.93	Normal	No	0.98	2.88	
5	0.90	Tumor	Yes	0.98	3.78	

of unweighted majority voting, each rule has the same weight, even though, it also has a differential score.

Here, we applied weighted majority voting. Let us assume that the class label of a rule is *normal*. If a sample satisfies the relation of the rule, the rule's score is added to the cumulative normal score; otherwise, it is added to the cumulative tumor score. For the final prediction, the class receiving the higher cumulative score is chosen. The weighted majority voting process is depicted in Table V.

C. Heuristics

The following heuristics are used to eliminate unnecessary rule combination, in order to ensure that a large amount of rules converge quickly to a *k*-high-scoring rule list, and to achieve high classification accuracy.

1) Combine rules with the same class label. If a rule-labeled *normal* is significant, S_N must be high, S_T must be low, and the score difference between S_N and S_T must be large. For a rule-labeled *tumor*, the converse is true. If two rules with different class labels are combined, the estimated maximum score of both S_N (Min(*p, q*)) and S_T (Min(*r, s*)) will be low, as shown in Table IV.
2) Keep the element list sorted by S_N (normal score) during combination among rules with the *normal* class label. Likewise, keep the element list sorted by S_T (tumor score) during combination among rules with the *tumor* class label.
3) The number of rules with *normal* class label is balanced with the number of rules with *tumor* class label. In most microarray experiments, it is hard to obtain equal numbers of normal and tumor tissues. Thus, microarray datasets have skewed samples with far fewer normal samples than tumor samples. If one uses such data for training datasets, S_N is usually greater than S_T and *normal* rules have a higher chance of being selected than *tumor* rules. Long *normal* rules such as Rule $= \{\text{Relation: } G_1 > G_2 > G_3 > G_4; \text{Class : Normal}\}$ are stringent when predicting whether an independent sample is *normal,* but lenient when predicting whether an independent sample is *tumor*. If such long *normal* rules are dominant in the classifier, the chance that Type-II errors are aggregated is increased because *k*-TSN is an ensemble method with multiple decision rules.
4) When there are rules with the same score, we select the rule that has the higher mean difference value considering the variance. In order to provide a statistical test of whether the means of several groups are all equal, we used Kruskal–Wallis one-way-analysis of variance by ranks test [29] that is a nonparametric method for testing the equality of means among groups when it cannot be assumed that the groups were selected from normal population. Each group in our algorithm is the set of rank values of a gene in samples with the same class label as the rule. If a rule consists of three genes, then there are three groups. The statistical significance of each rule is tested for by comparing the *H*-test statistics to the chi-square distribution with (*number of groups* -1) degrees of freedom.

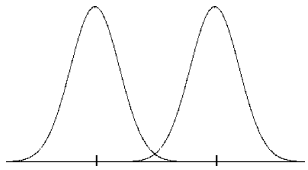

Fig. 5. Distribution of rank values for distant genes G_i and G_j.

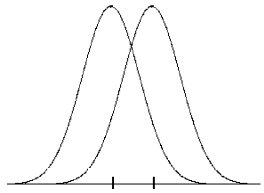

Fig. 6. Distribution of rank values for close genes G_i and G_j.

$$H = \frac{12}{n(n+1)} \sum_{i=1}^{g} \frac{R_i^2}{n_i} - 3(n+1)$$

where *g* is the number of groups, *n* is the *g* (*number of samples with the same class label as the rule*), R_i^2 is the sum of the rank values corresponding to the *i*th group, and n_i is the number of samples for group *i*. Here, n_i is same for all the groups, which is *number of samples with the same class label as the rule*.

We calculated the *p*-value by $\Pr(\chi_{g-1}^2 \geq h)$ for each rule and selected the most significant rule, which has the smallest *p*-value among the rules with the same score. Figs. 5 and 6 show the distributions of the rank values for genes G_i and G_j in a set of samples with the same class label as the rule. The distribution in Fig. 5 shows a more differentially expressed gene relation for G_i and G_j.

D. Algorithm Summary

This section illustrates the overall *k*-TSN algorithm. *k-TSN* (see Fig. 7) is the main algorithm. It makes pair rules, separates them by the class label, and joins them to make triple, quadruple, or higher rules. *MakePairRule* (see Fig. 8) generates a pair-rule list. It generates all possible pairs of genes and computes scores for all possible pair rules. *Join* (see Fig. 9) is a central algorithm. It estimates the score range of the new longer rule and decides whether or not to actually calculate the score of the longer rule. If the estimated maximum score is greater than the minimum score

k-TSN Algorithm
Input: a Learning Dataset(LS), k
Output: k-Rule List
1. PairRule ← MakePairRule(LS).
2. Initialize 2 Empty Arrays: NormalPairRules and TumorPairRules.
3. Repeat for each rule R in PairRule
4. If R is Normal Rule Then insert R into NormalPairRules.
5. Else insert R into TumorPairRules.
End
6. NormalRules←Join(NormalPairRules,k/2,Normal).
7. TumorRules←Join(TumorPairRules,k/2,Tumor).
8. Return the result of 6 and 7.

Fig. 7. *k*-TSN algorithm.

MakePairRule Algorithm
Input: Learning Dataset(LS) with S Samples, P Genes
Output: PairRuleList
1. PairRuleList ← Initialize an Empty Array.
2. For i=1:P
3. For j=1:P and j≠i
4. R←Make a Rule{Relation : $G_i > G_j$;}
/* G_i, G_j means Gene i, Gene j */
5. Compute NS(normal score) and TS(tumor score).
6. If NS > TS then set the Class as Normal.
7. Else set the Class as Tumor.
8. Set the Score as |NS-TS|.
9. Insert R into PairRuleList.
End
End
10. Return PairRuleList.

Fig. 8. MakePairRule algorithm.

of the current *k*-rule list, it calculates the real score. Otherwise, it skips the calculation to reduce computation. Once the real score is calculated, it inserts this new rule into the *k*-rule list if the calculated score is greater than the minimum score of the current *k*-rule list, and the minimum score is updated. It inserts this new rule into the element list if the calculated primary score is greater than the minimum score of the current *k*-rule list. *ChkAddable* (see Fig. 10) checks the gene in the ending position of an ahead rule and the gene in the beginning position of a behind rule, and returns true if they are same. *Combine* (see Fig. 11) actually calculates the real score of a new rule and elongates the shorter rules into a longer rule.

IV. Experimental Results

In this section, we demonstrate the efficacy of the *k*-TSN method for several gene expression datasets involving prostate and colon cancers, by comparing the run times and classification accuracies with those of *k*-TSP and *k*-TST.

In the previous work [7], we made an extensive comparison study of three-gene rule case, *k*-TST with a current method such as SVM. The experimental results of *k*-TST show that it did not always perform better than SVM when the training dataset was a single microarray data having small number of samples and skewed samples with far fewer normal samples than tumor

Join Algorithm
Input: PairRuleList, k(number of rules to be selected), Rule Type(Normal or Tumor)
Output: k-Rule List
1. k-Rule List ← Initialize an Empty Array.
2. Repeat for k times
3. Select the best (High Score) Pair Rule.
4. Insert the Rule into k-Rule List.
5. Remove the Rule from PairRuleList.
End
6. minScore ← the lowest Score from k-Rule List.
7. If Rule Type = Normal then PrimaryScore ← NS, SecondaryScore ← TS.
8. Else PrimaryScore ← TS, SecondaryScore ← NS.
9. ElementList ← Sort PairRuleList by PrimaryScore.
10. While ElementList is not empty
11. Remove Rules from ElementList that have lower PrimaryScore than minScore.
12. If ElementList is empty then return k-Rule List.
13. HighRules ← Select the Rules (possibly multiple) that have highest PrimaryScore from ElementList.
14. LowRules ← Select all Rules from ElementList except the rules in HighRules.
15. Delete Rules in 13 from ElementList.
16. For each HR in HighRules
17. For each LR in LowRules
18. If ChkAddable(HR,LR) = false then go to 17.
19. UpperBound = min(HR.PrimaryScore, LR.PrimaryScore).
20. LowerBound = max(HR.SecondaryScore + LR.SecondaryScore-1, 0).
21. If UpperBound-LowerBound < minScore then go to 17.
22. JoinR ← Combine(HR, LR).
23. If JoinR.Score > minScore then
24. Insert JoinR into k-Rule List.
25. Erase a rule that has the lowest Score from k-Rule List.
26. Update minScore.
End
27. If JoinR.PrimaryScore > minScore
28. Insert JoinR into ElementList.
End
End // For each LR in LowRules
End // For each HR in HighRules
End // While ElementList is not empty
29. Return k-Rule List.

Fig. 9. Join algorithm.

ChkAddable Algorithm
Input: Two Rules(RuleA, RuleB)
Output: True or False
1. If RuleA = RuleB then return false.
2. If RuleA.FirstGene = RuleB.LastGene (or RuleB.FirstGene = RuleA.LastGene) then Return true.
3. Return false.

Fig. 10. ChkAddable algorithm.

Combine Algorithm
Input: Two Rules(RuleA, RuleB)
Output: CombinedRule
1. If RuleA.FirstGene = RuleB.LastGene then
   BeginningR = RuleB, EndingR = RuleA.
2. Else BeginningR = RuleA, EndingR = RuleB.
3. CombinedRule ← BeginningR.
4. Elongate CombinedRule.Relation with
   EndingR.Relation.
5. Compute CombinedRule.NS and CombinedRule.TS.
6. If NS > TS, then Set CombinedRule.Class as Normal.
7. Else Set CombinedRule.Class as Tumor.
8. Set CombinedRule.Score as |NS-TS|.
9. Return CombinedRule.

Fig. 11. Combine algorithm.

TABLE VI
PROSTATE MICROARRAY DATA

Data	Number of Genes	Number of Normal Samples	Number of Tumor Samples	Total Number of Samples
Welsh	12626	9	24	33
LaTulippe	12626	3	23	26
Singh	12600	50	52	102

samples. However, as integration increases the sample size, *k*-TST performed with better accuracy than the SVM method. For the three respective integrated Affymetrix microarray datasets, the accuracies of the *k*-TST are 85.29%, 96.97%, and 100%, while the accuracies of SVM method are 58.82%, 87.88%, and 100%. For the two respective integrated cDNA microarray datasets, the accuracies of the *k*-TST are 97.67% and 97.16%, while the accuracies of the SVM method are 83.72% and 91.47%. Accordingly, we focused on comparing the proposed method with *k*-TST for this experiment. For *k*-TSP, we use the executables provided by Tan [27], and for *k*-TST, we use those provided by Yoon [7].

We use the publicly available prostate cancer microarray data whose platform is the Affymetrix HG_95AV2 [30]. These are oligonucleotide microarrays. For the sake of convenience, each dataset is represented by the surname of the first author of the papers by: Singh *et al.* [31], Welsh *et al.* [32], and LaTulippe *et al.* [28].

We use another set of microarray data from a colon cancer study at the Cancer Metastasis Research Center of Yonsei University, whose platform is cDNA. These are spotted microarrays. The colon cancer datasets are preprocessed with a nonmissing proportion (NMP) of 100%. Tables VI and VII show the details of the microarray datasets used in our experiment.

A. Optimal k-Selection by LOOCV

Every training dataset has its own optimal *k*-value. The parameter *k* in the *k*-TSN is determined by the standard LOOCV, while *k* is increased from 5 to 15. The optimal value of *k* and the LOOCV accuracy are illustrated in Table VIII. The optimal *k*-value for *k*-TSP and *k*-TST is derived from [7].

TABLE VII
COLON CANCER CDNA MICROARRAY DATA

Data Name	Number of Genes	Number of Normal Samples	Number of Tumor Samples	Total Number of Samples	Characteristics
A_batch_Paired	8125	131	131	262	Batch A, Paired, Used as Training Data
A_batch_Unpaired	10652	0	86	86	Batch A, Unpaired, Used as Training & Test Data
B_batch_Unpaired	12855	0	211	211	Batch B, Unpaired, Used as Training & Test Data

TABLE VIII
OPTIMAL k-VALUE AND LOOCV ACCURACY FOR k-TSN

Training Dataset	Best K	Accuracy(%)
Welsh	11	93.94
Latulippe	11	100
Singh	15	92.2
Welsh+Latulippe	5	96.61
Welsh+Singh	7	91.85
Latulippe+Singh	9	90.63
A batch Paired	15	98.09
A batch Paired+A batch Unpaired	15	96.83
A batch Paired+B batch Unpaired	9	95.98

B. Stability of k-TSN Classifier

Stability is defined as the sensitivity of our algorithm to variations in the training set. We quantify how different subsamples of a training set affect the composition of the decision rules. We used the Kuncheva index [33] to measure the stability. The stability index for a set of classifiers of decision rules, for a given set size *k*, is the average of all pair-wise consistency indexes. We measured the stability index from the set of classifiers from each LOOCV run. If the Singh data are used for LOOCV, the number of runs is 102, which is equal to the number of samples. We measured the stability index for a set of 102 classifiers of decision rules. In Fig. 12, stability index of two training datasets for every *k* value are shown. In both datasets, the optimal *k*-value was 15. For *k*-value of 15, the stability index of Singh is 0.7593, and the stability index of A_batch_paired is 0.9053.

$$I_s(k) = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} I_c(S_i(k), S_j(k))$$

where *N* is the number of samples and $k = 5, 7, 9, 11, 13, 15$.

The pair-wise consistency index for the two rule sets *A* and *B* is defined as follows:

$$I_c(A, B) = \frac{rn - k^2}{k(n-k)}$$

where *r* is the cardinality of the intersection of the two sets: *A* and *B* and *n* is the number of rules in *A* and *B*.

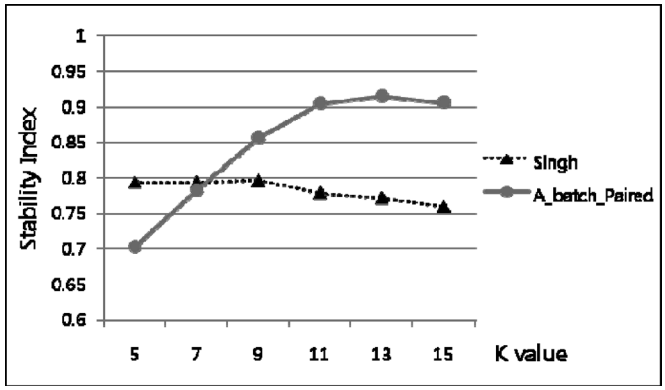

Fig. 12. Stability index of *k*-TSN classifier.

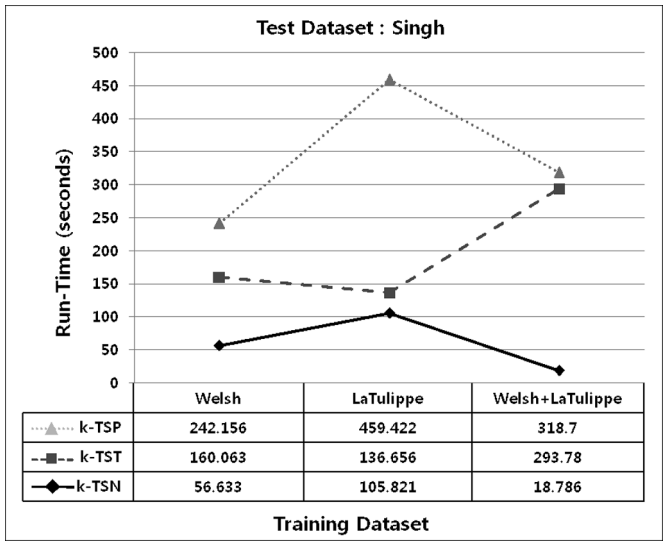

Fig. 13. Run time for Singh test dataset.

C. Run Time

We make a comparative study of *k*-TSP, *k*-TST, and *k*-TSN for the run time. For *k*-TST and *k*-TSN, the informative gene selection stage is run first, and we only use 1% informative genes out of original genes for classification.

Figs. 13–15 show a comparison of the run times for the prostate cancer datasets. The figures show that *k*-TSN runs faster than the other two algorithms. For *k*-TSN, the converging rate of the *k*-rule list depends on the training dataset. It takes a relatively longer time to build a *k*-rule list for a dataset such as LaTulippe, which has a small number of samples and also has skewed samples with far fewer normal samples than tumor samples. However, there is rapid convergence toward the high-scoring rule list for a dataset such as Singh, which has a large sample size, and a balanced number of normal and tumor samples. It takes less time when Singh is used as a training dataset than when Latulippe is used as training dataset.

For integrated microarray datasets using the rank values, *k*-TST requires more computation time with an increased number of samples by integration. However, *k*-TSN requires less time for integrated datasets, because integration ensures that the sample

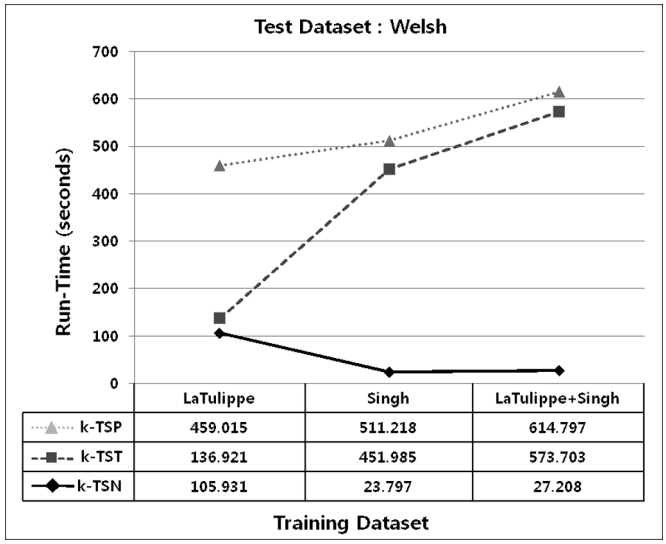

Fig. 14. Run time for Welsh test dataset.

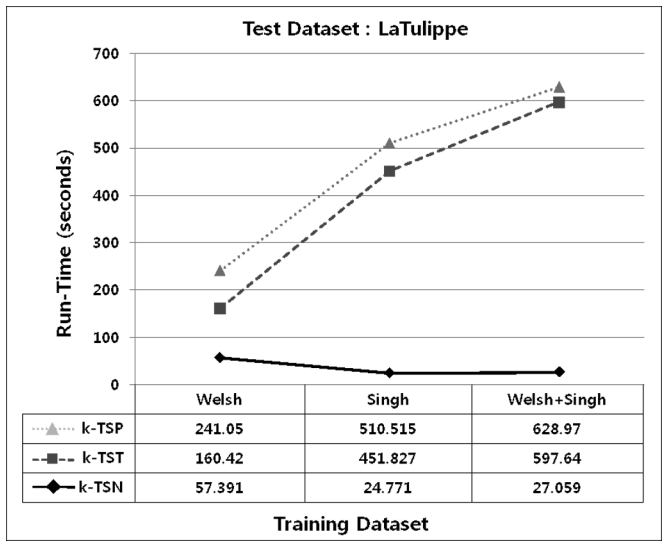

Fig. 15. Run time for Latulippe test dataset.

size of a training dataset is larger, and the number of normal and tumor samples is balanced. This results in rapid convergence toward the high-scoring rule list.

Figs. 16 and 17 pertain to the colon cancer microarray datasets with the cDNA platform and large sample sizes of more than 200. They show results similar to those of the Affymetrix prostate cancer microarray datasets. *k*-TSN shows drastically less time for all three training datasets, since the training datasets have large sample sizes.

k-TST shows an increase in run time that is proportional to the sample size of the training dataset. However, *k*-TSN shows a decrease in run time for the integrated datasets. A comparison of run times reveals that *k*-TSN runs between 4 and 20 times faster than *k*-TSP (worst to best). In comparison with *k*-TST, *k*-TSN runs between 1.2 and 20 times faster (worst to best).

It is highly expected and experimentally observed that the iteration does not proceed further above fifth or higher for

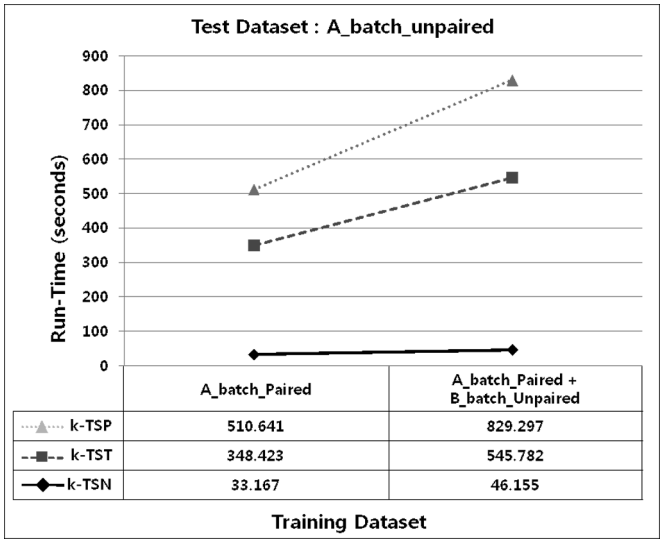

Fig. 16. Run time for A_batch_unpaired test dataset.

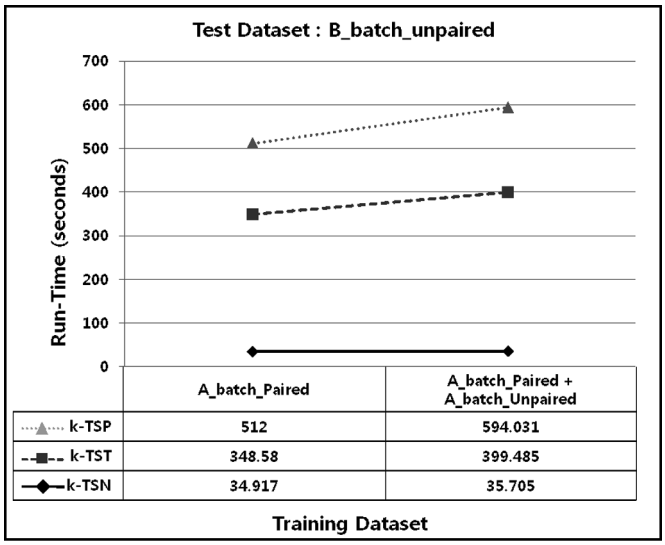

Fig. 17. Run time for B_batch_unpaired test dataset.

microarray datasets that have balanced number of normal and tumor samples. As the iteration proceeds, a new longer rule is generated and as the rule is getting longer, the samples rarely satisfy the relation of longer rule, and accordingly, the primaryScore of the newly built rule is hardly high. However as the iteration proceeds, the current minScore of the *k*-rule list is getting higher. Since the element list keeps only the rules, whose primaryScore is higher than the minScore, the number of the rules in the element list is decreasing drastically and the rules in the element list are exhausted, as the iteration proceeds.

D. Classification Accuracy

In this section, we train the *k*-TSN classifiers on the training datasets, evaluate their performance on the independent test datasets, and compare the results with those of random forest, *k*-TSP, and *k*-TST.

TABLE IX
TOTAL NUMBER OF GENES IN A *K*-TSN CLASSIFIER FOR EACH TRAINING DATASET

Training Dataset	Best K	Total Number of Genes
Welsh	11	36
Latulippe	11	68
Singh	15	50
Welsh+Latulippe	5	20
Welsh+Singh	7	20
Latulippe+Singh	9	29
A batch Paired	15	36
A batch Paired+A batch Unpaired	15	44
A batch Paired+B batch Unpaired	9	24

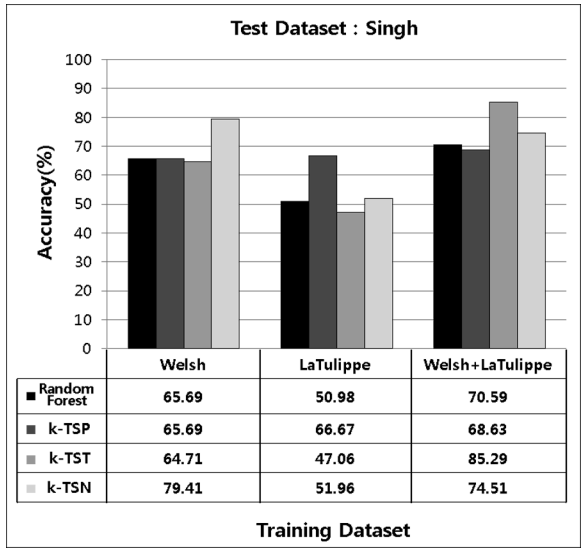

Fig. 18. Accuracy for Singh test dataset.

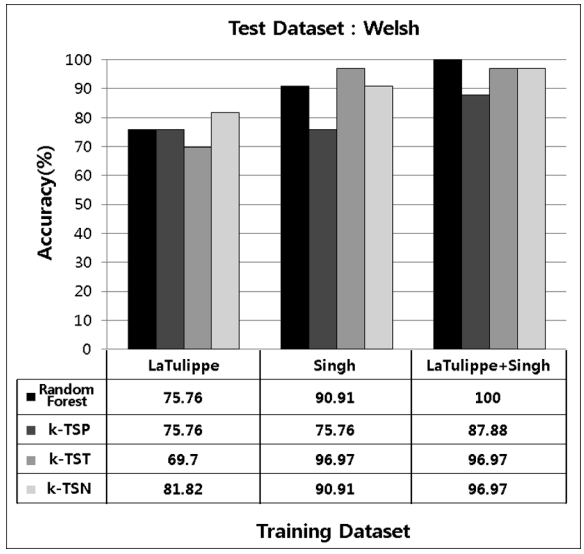

Fig. 19. Accuracy for Welsh test dataset.

First, the total number of genes in a *k*-TSN classifier for each training dataset is shown in Table IX.

When a training dataset has few and unbalanced samples, as in Welsh, Latulippe, and the integrated dataset of Welsh +

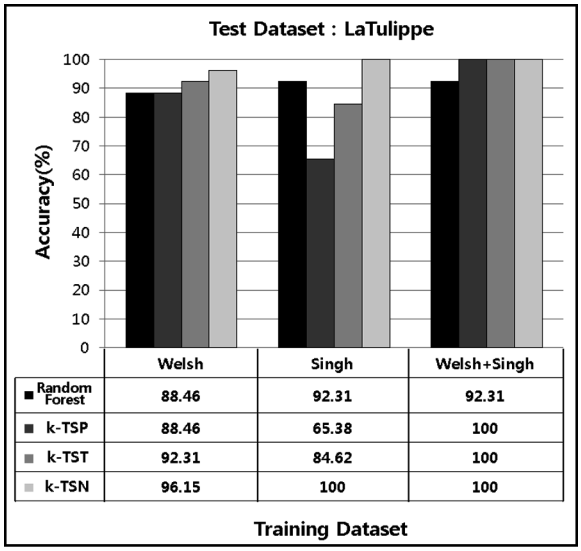

Fig. 20. Accuracy for Latulippe test dataset.

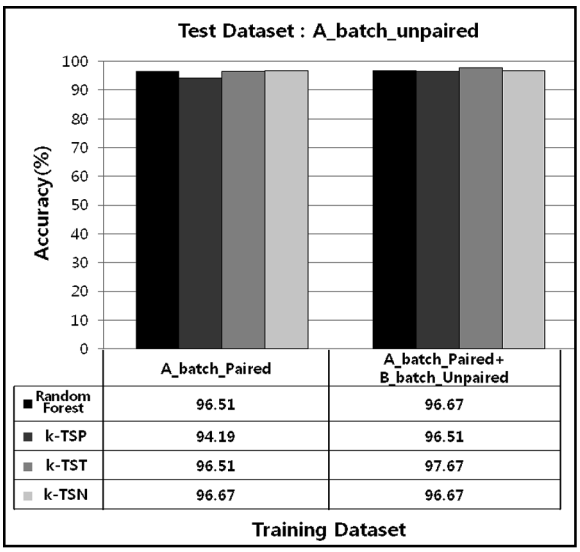

Fig. 21. Accuracy for A_batch_unpaired test dataset.

Latulippe, all four algorithms show poor performance, as shown in Fig. 18. For larger training datasets such as Singh, Singh + Welsh, and Singh + Latulippe, as shown in Figs. 19 and 20, k-TSN is consistently more accurate than k-TSP and has comparable accuracy to k-TST. As long as a training dataset secures a sufficient sample size, k-TSN shows better performance than k-TSP and comparable performance to k-TST.

Figs. 21–22 show the classification accuracy for cDNA colon cancer datasets. All four algorithms perform well in this case because the sample sizes are sufficiently large.

For a training dataset of a large sample size ($\geq$100 samples), k-TSN achieved a high accuracy rate, with an increase of up to 11%, with respect to k-TSP, and a comparable accuracy to k-TST. In seven out of nine runs using the prostate cancer datasets, the k-TSN method offers a slightly higher or comparable accuracy in comparison with k-TST. k-TSN consistently

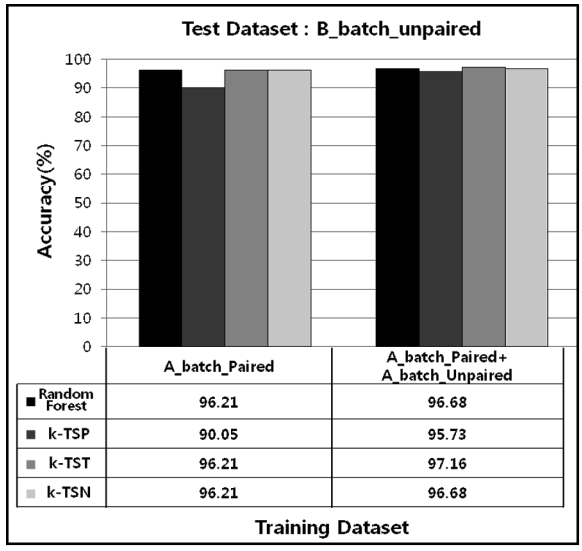

Fig. 22. Accuracy for B_batch_unpaired test dataset.

achieves a high classification accuracy for the test datasets from the two most commonly used microarray platforms (Affymetrix and cDNA).

V. Conclusion

We proposed a novel classification method k-TSN that generalizes the number of genes in a decision rule and compared the experimental results of k-TSP, k-TST, and k-TSN. The proposed classifier is an ensemble method with a biologically intuitive interpretation. It is data-driven machine learning method composed of k-decision rules. Our approach builds a new longer rule by combining shorter rules of which scores were already calculated while estimating the score range of a new rule. It stops considering the new rule if the estimated maximum score is below the current minimum in the rule list. This differs from current (k-TSP and k-TST) methods that generate all possible rules of gene combinations from a training dataset and calculate the scores for all of the rules. In comparison, the proposed method saves computing time and memory space.

For the training datasets, we used a single microarray dataset, and the rank-based integrated microarray datasets. We made a run-time comparison of k-TSN with k-TSP and k-TST. In particular, k-TSN runs faster than the current methods when the training datasets are integrated microarray datasets with large sample sizes. k-TSN shows higher accuracy than k-TSP and comparable accuracy to k-TST for training dataset of a sufficient sample size ($\geq$100 samples).

The k-TSN classifier provides a fast run time, high classification accuracy, and a biologically intuitive interpretation. It can identify exactly which genes are involved in the classifier. Since the number of genes involved in k-TSN classifier is limited, this suggests a potential use in a clinical setting.

References

[1] S. Mitra and Y. Hayashi, "Bioinformatics with soft computing," *IEEE Trans. Syst., Man, Cybern. C*, vol. 36, no. 5, pp. 616–635, Sep. 2006.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2006.
[3] C. Campbell, S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, T. R. Golub, and J. Mesirov, "Estimating dataset size requirements for classifying DNA microarray data," *J. Comput. Biol.*, vol. 10, pp. 119–142, 2003.
[4] Y. Lai, B. Adam, R. Podolsky, and J. She, "A mixture model approach to the tests of concordance and discordance between two large-scale experiments with two-sample groups," *Bioinformatics*, vol. 23, pp. 1243–1250, 2007.
[5] Y. Lu and J. Han, "Cancer classification using gene expression data," *Inf. Syst.*, vol. 28, pp. 243–268, 2003.
[6] M. Banerjee, S. Mitra, and H. Banka, "Evolutionary rough feature selection in gene expression data," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 4, pp. 622–636, Jul. 2007.
[7] Y. Yoon, J. Lee, S. Park, S. Bien, H. C. Chung, and S. Y. Rha, "Direct integration of microarrays for selecting informative genes and phenotype classification," *Inf. Sci.*, vol. 178, pp. 88–105, 2008.
[8] L. M. Fu and C. S. Fu-Liu, "Evaluation of gene importance in microarray data based upon probability of selection," *BMC Bioinf.*, vol. 6, pp. 67–77, 2005.
[9] X. X. Liu, A. Krishnan, and A. Mondry, "An entropy-based gene selection method for cancer classification using microarray data," *BMC Bioinf.*, vol. 6, pp. 76–89, 2005.
[10] E. K. Tang, P. N. Suganthan, and X. Yao, "Gene selection algorithms for microarray data based on least square support vector machine," *BMC Bioinf.*, vol. 7, pp. 95–110, 2006.
[11] W. E. Johnson, C. Li, and A. Rabinovic, "Adjusting batch effects in microarray expression data using empirical Bayes methods," *Biostatistics*, vol. 8, pp. 118–127, 2007.
[12] G. Bloom, I. V. Tang, D. Boulware, K. Y. Kwong, D. Coppola, S. Eschrich, J. Quackenbush, and T. J. Yeatman, "Multi-platform, multisite, microarray-based human tumor classification," *Amer. J. Pathol.*, vol. 164, pp. 9–16, 2004.
[13] J. Khan, J. Wei, M. Ringnér, L. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. Antonescu, C. Peterson, and P. Meltzer, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nat. Med.*, vol. 7, no. 6, pp. 673–679, 2001.
[14] E. Wit and J. McClure, *Statistics for Microarrays: Design, Analysis, and Inference*. Hoboken, NJ: Wiley, 2004.
[15] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learning*, vol. 46, pp. 389–422, 2002.
[16] T. Joachims. (2003). *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms* [Online]. Norwell, MA: Kluwer. Available: http://svmlight.joachims.org/
[17] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVM," in *Proc. Neural Inf. Process. Syst. Conf. (NIPS)*, 2000, pp. 668–674.
[18] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learning*, vol. 46, pp. 131–159, 2002.
[19] I. Guyon and A. Elisseef, "An introduction to variable and feature selection," *J. Mach. Learning Res.*, vol. 3, pp. 1157–1182, 2003.
[20] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1993.
[21] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2005.
[22] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learning*, vol. 40, pp. 139–157, 2000.
[23] A. C. Tan and D. Gilbert, "Ensemble machine learning on gene expression data for cancer classification," *Appl. Bioinf.*, vol. 2, pp. 75–83, 2003.
[24] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, pp. 5–32, 2001.
[25] Long and P. M. and V. B. Vega, "Boosting and microarray data," *Mach. Learning*, vol. 52, pp. 31–44, 2003.
[26] Y. Saeys, T. Abeel, and Y. Van de Peer, "Robust feature selection using ensemble feature selection techniques," in *Proc. ECML/PKDD, Part II (LNAI 5212)*, 2008, pp. 313–325.
[27] A. Tan, D. Naiman, L. Xu, R. Winslow, and D. Geman, "Simple decision rules for classifying human Cancers from gene expression profiles," *Bioinformatics*, vol. 21, pp. 3896–3904, 2005.
[28] E. LaTulippe, J. Satagopan, A. Smith, H. Scher, P. Scardino, and V. Reuter, "Comprehensive gene expression analysis of prostate Cancer reveals distinct transcriptional programs associated with metastatic disease," *Cancer Res.*, vol. 62, pp. 4499–4506, 2002.
[29] R. Walpole, R. Myers, and K. Ye, *Probability & Statistics for Engineers & Scientists*, 7th ed. Delhi, India: Pearson Education, 2002.
[30] (2009). [Online]. Available: http://www.affymetrix.com/index.affx
[31] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, and C. Ladd, "Gene expression correlates of clinical prostate Cancer behavior," *Cancer Cell*, vol. 1, pp. 203–209, 2002.
[32] J. B. Welsh, L. M. Sapinoso, A. I. Su, S. G. Kern, J. Wang-Rodriguez, and C. A. Moskaluk, "Analysis of gene expression identifies candidate markers and pharmacological targets in prostate Cancer," *Cancer Res.*, vol. 61, pp. 5974–5978, 2001.
[33] L. Kuncheva, "A stability index for feature selection," in *Proc. 25th Conf. IASTED Int. MultiConf. (AIAP 2007)*, Anaheim, CA: ACTA Press, pp. 390–395.

Youngmi Yoon received the B.S. degree in microbiology from Seoul National University, Seoul, Korea, in 1981 and completed all undergraduate requirements for mathematics major at the Ohio State University, Colombus, in 1983, the M.S. degrees in statistics and computer science from Stanford University, Standford, in 1984, and 1987, respectively, and the Ph.D. degree in computer science from Yonsei University, Seoul, in 2008, under the supervision of Prof. S. Park.

For five and half years, she worked as a Software Engineer in IntelliGenetics, Inc., Mountain View, CA. She is currently a Professor of Information Technology Division, Gachon University of Science and Medicine, Icheon, Korea, where she was an Assistant Professor in 1995. Her research interest includes database, data mining, and bioinformatics.

Sangjay Bien received the B.S. degree in computer science and biotechnology from Yonsei University, Seoul, Korea. He is currently working toward the master's degree with Biomedical Informatics Laboratory, Department of Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul, where he is engaged in semantics of genes using each GO terms, and measures similarity and distance of known genes.

He studied classification of microarray with S. Park and Ms. Y. Yoon since 2006. His current research interests include functional annotation of genes and gene ontology, comparative microarray data analysis, interpretation of genotype, and discovery of novel genomic function.

Sanghyun Park received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, Korea, in 1989 and 1991, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2001, under the supervision of Prof. W. W. Chu.

He is currently an Associate Professor with the Department of Computer Science, Yonsei University, Seoul. His research interests include database, data mining, and bioinformatics.