BIOINFORMATICS

Vol. 19 no. 16 2003, pages 2097–2104
DOI: 10.1093/bioinformatics/btg288

Kernel hierarchical gene clustering from microarray expression data

Jie Qin[1,*], Darrin P. Lewis[2] and William Stafford Noble[3,†]

[1]Columbia Genome Center, Columbia University, 1150 St. Nicholas Avenue, New York, NY 10032, USA, [2]Department of Computer Science, Columbia University, 1214 Amsterdam Avenue, New York, NY 10027, USA and [3]Department of Genome Sciences, University of Washington, 1705 NE Pacific Street, Seattle, WA 98195, USA

Received on July 13, 2002; revised on December 20, 2002; accepted on May 8, 2003

ABSTRACT

Motivation: Unsupervised analysis of microarray gene expression data attempts to find biologically significant patterns within a given collection of expression measurements. For example, hierarchical clustering can be applied to expression profiles of genes across multiple experiments, identifying groups of genes that share similiar expression profiles. Previous work using the support vector machine supervised learning algorithm with microarray data suggests that higher-order features, such as pairwise and tertiary correlations across multiple experiments, may provide significant benefit in learning to recognize classes of co-expressed genes.

Results: We describe a generalization of the hierarchical clustering algorithm that efficiently incorporates these higher-order features by using a kernel function to map the data into a high-dimensional feature space. We then evaluate the utility of the kernel hierarchical clustering algorithm using both internal and external validation. The experiments demonstrate that the kernel representation itself is insufficient to provide improved clustering performance. We conclude that mapping gene expression data into a high-dimensional feature space is only a good idea when combined with a learning algorithm, such as the support vector machine that does not suffer from the curse of dimensionality.

Availability: Supplementary data at www.cs.columbia.edu/compbio/hiclust. Software source code available by request.

Contact: jq22@columbia.edu

INTRODUCTION

Finding structure in large data sets is a venerable, well-studied problem that has recently received an explosion of interest from biologists using microarray expression measurement technology. Eisen *et al*. (1998) popularized the use of hierarchical clustering (Duda and Hart, 1973) to find groups of similarly expressed genes or gene expression experiments. This algorithm has several well-known drawbacks, most notably that it is statistically unstable in the face of small perturbations of the data. Therefore, many subsequent papers have suggested applying alternate clustering algorithms to microarray data. Some of these algorithms were adopted from prior literature in other fields and some were developed specifically for microarray data (see Slonim, 2002, for a review). However, the standard hierarchical clustering algorithm remains among the most widely used in this field, due to its simplicity, its intuitive, tree-structured output and the availability of several free software tools.

Concurrently, within the field of machine learning, much recent research has focused upon a class of algorithms that employ so-called *kernel functions* to operate efficiently on a non-linearly transformed version of a given data set. The motivation for this transformation is simple: the features, or variables, within a given data set often share complex, non-linear relationships. Say that you are looking for a correlation between gene expression levels and some clinical variable. It may be the case that no single gene expression level is directly related to the clinical variable of interest, but a pair of genes is strongly predictive (see Table 1). In this situation, the product of the expression levels of the two genes is much more informative than the expression level of either gene by itself. A simple solution is to multiply all pairs of gene expression values, and operate on this transformed data set. However, when an expression data set contains 10 000 genes, looking at all pairs of genes results in a data set of 100 000 000 genes. Analyzing such a large data set is computationally expensive, and only gets worse if we are interested in tertiary, rather than pairwise, correlations.

The kernel trick allows higher-order features, such as pairwise or higher correlations, to be analyzed efficiently. Any algorithm can be 'kernelized' if the algorithm can be stated so that each vector of input data only appears within a dot product operation. The non-linear version of the algorithm is then created by replacing the normal dot product with an alternate function, known as a kernel function. Mercer's

*To whom correspondence should be addressed.
†Formerly William Noble Grundy, see www.gs.washington.edu/~noble/name-change.html

Table 1. A non-linear relationship between gene expression level and a clinical variable

	Experiment 1	2	3	4
Gene 1	Low	High	Low	High
Gene 2	Low	Low	High	High
Clinical variable	False	True	True	False

Neither gene's expression level is sufficient to predict the Boolean variable, but the two expression levels together are predictive.

theorem (Cristianini and Shawe-Taylor, 2000) guarantees that, as long as the kernel function exhibits certain mathematical properties (namely, that it is positive definite), then the algorithm implicitly operates in a higher-dimensional space. For example, if the standard dot production operation is replaced by the square of the dot product, i.e. if $K(X, Y) = (X \cdot Y)^2$, then we can prove that the kernelized algorithm, although operating only on 10 000 gene features, will yield the same results as if the non-kernelized algorithm were given all 100 000 000 pairwise products of the gene features. The kernel trick saves the algorithm from the computational expense of explicitly representing all of the features in the higher-dimensional space.

The kernel trick was popularized in the context of support vector machine (SVM) learning (Vapnik, 1998). The SVM is a supervised machine learning algorithm that is frequently used in a kernelized form. Subsequently, kernel versions of several other algorithms have been described, including Fisher's linear discriminant (Mika *et al*., 1999) and principal components analysis (Schölkopf *et al*., 1997). The current work is particularly motivated by a study that demonstrated the applicability of the SVM algorithm to learning gene classifications from microarray expression data (Brown *et al*., 2000). In this paper, the classification performance of the SVM improved considerably when a non-linear kernel function was inserted into the algorithm. This result suggests that gene expression data contains informative, higher-order features which may not be apparent in the raw data. Such relationships are not surprising, given the complex network of gene regulatory interactions that give rise to the observed expression data.

In this paper, therefore, we demonstrate how to apply the kernel trick to the hierarchical clustering algorithm. The resulting algorithm is nearly as efficient as standard hierarchical clustering, and can operate in very high-dimensional, implicit feature spaces. We then empirically validate the utility of the resulting algorithm on the task of clustering genes based upon the microarray expression profiles. The results are not encouraging. In short, although kernel hierarchical clustering does produce different results from standard hierarchical clustering, those results are not necessarily any better. Using multiple data sets, we evaluate the goodness of the clustering results internally, by predicting the cluster membership of held-out examples, as well as externally, by comparing clusters with functional annotations. Using either measure, the kernel-derived clusters are sometimes better and sometimes worse than the standard clusters. This work is therefore a cautionary tale. We hypothesize that the advantage offered by the kernel trick may be fairly specific to the SVM and other large margin classifiers that do not suffer dramatically from the curse of dimensionality.

ALGORITHM

Standard hierarchical clustering

Agglomerative hierarchical clustering consists of initially treating each input point as a cluster and then iteratively merging the two clusters that are closest together. This iterative merging procedure continues until all points have been merged into a single cluster. The order of mergers determines the topology of a binary tree, which is the output of the algorithm. Clusters can be derived by selecting subtrees from this tree.

The primary difference among hierarchical clustering algorithms lies in the method of finding the closest pair of clusters in line (3) of Figure 1. In this paper, we consider four methods of measuring similarities among clusters. All are based on a common metric for measuring the similarity of a pair of points. In many applications, this point-wise comparison is based upon a Euclidean distance. However, for comparing gene expression profiles, Eisen *et al*. (1998) employ a modified version of the Pearson correlation coefficient rather than the Euclidean distance. This modified correlation $r(\cdot, \cdot)$ between X and Y is

$$\hat{r}(X, Y) = \frac{\sum X_i Y_i}{\sqrt{\left(\sum X_i^2\right)\left(\sum Y_i^2\right)}} \qquad (1)$$

Using this metric, the similarity between a pair of clusters $\mathcal{A}$ and $\mathcal{B}$ can be computed in four ways. In single-link clustering, the similarity between $\mathcal{A}$ and $\mathcal{B}$ is simply the maximum similarity between a point in cluster $\mathcal{A}$ and a point in cluster $\mathcal{B}$. Complete-link clustering is the same, but using the minimum point-wise similarity. Average-link clustering, as the name suggests, uses the mean of all pairwise similarities between clusters. The fourth algorithm, centroid-link clustering, computes the distance between clusters by first representing each cluster via a centroid or mean vector, each component of which is the mean of the corresponding components across all vectors in the cluster. The inter-cluster similarity is the similarity of these two mean vectors, computed according to Equation (1).

In the first three of these four clustering algorithms, the merging of two clusters [line (4) in Fig. 1] is trivial. For the centroid-link algorithm, the merging can be accomplished efficiently by combining previously computed mean

```
(1)  Initialize every point as a cluster
(2)  while more than one cluster remains
(3)      Find the closest pair of clusters
(4)      Merge the two clusters
(5)  end
```

Fig. 1. Generic agglomerative hierarchical clustering algorithm. The differences among hierarchical clustering algorithms lie in the methods of calculating the distance between clusters in line (3).

vectors. Given two such vectors, $A = A_1, A_2, \ldots, A_n$ and $B = B_1, B_2, \ldots, B_n$, representing clusters of sizes a and b, respectively, the algorithm computes the new mean vector $\hat{A}$ by taking the weighted mean of each element:

$$\hat{A}_i = \frac{aA_i + bB_i}{a+b} \tag{2}$$

Kernel hierarchical clustering

The kernel version of hierarchical clustering uses the kernel trick to map implicitly the gene expression data into a higher-dimensional feature space. Clustering then takes place in the feature space. Note that the function Φ that maps from the input space to the feature space may not be known, but such a function is guaranteed to exist for any valid kernel function. For centroid-link hierarchical clustering, this implicit mapping occurs prior to the computation of centroids. The centroid of a cluster in feature space is not the same as the mapped version of the centroid computed in the input space.

In order to produce a kernel version of the hierarchical clustering algorithm, we must be able to state the entire algorithm in terms of dot products between pairs of input vectors. This transformation is straightforward for both the Euclidean distance (results not shown) and the modified Pearson correlation. The modified Pearson correlation [Equation (1)] can be re-stated in terms of dot products as follows:

$$\hat{r}(X, Y) = \frac{X \cdot Y}{\sqrt{(X \cdot X)(Y \cdot Y)}} \tag{3}$$

The kernel version of the algorithm employs Equation (3), with the kernel function $K(\cdot, \cdot)$ substituted in place of each dot product operation.

In addition to computing distances or correlations, the kernel version of centroid-link hierarchical clustering requires that the cluster merging operation be carried out in terms of dot product operations. We show inductively how this update step works. The base case occurs prior to any merging, when the kernel matrix K is computed for the entire data set. Each vector in the data set is a cluster of size 1, and the vector itself is trivially equal to the centroid of this single-vector cluster. For the inductive step, we assume that the kernel matrix K contains kernel values between pairs of cluster means. Let cluster $\mathcal{A}$, represented (in the feature space) by centroid vector $\Phi(A)$, consist of a vectors, and similarly

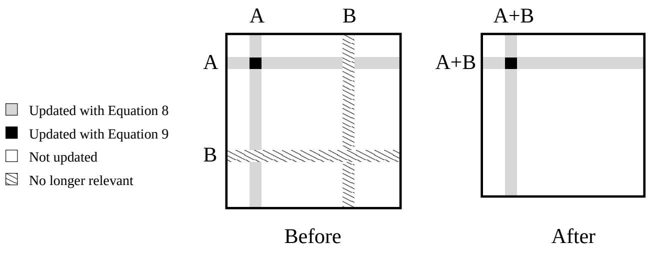

Fig. 2. Kernel update procedure.

for cluster $\mathcal{B}$. Then the corresponding kernel matrix entry is $K(\mathcal{A}, \mathcal{B}) = \Phi(A) \cdot \Phi(B)$. We want to merge $\mathcal{A}$ and $\mathcal{B}$ to form a new cluster $\hat{\mathcal{A}}$ containing $a + b$ vectors.

To update the kernel matrix, we need to deal with two cases. The first is the kernel value between the merged cluster $\hat{\mathcal{A}}$ and some other cluster $\mathcal{C}$. Following Equation (2), we can write the desired kernel value in terms of known kernel values:

$$K(\hat{\mathcal{A}}, \mathcal{C}) = \frac{aK(\mathcal{A}, \mathcal{C}) + bK(\mathcal{B}, \mathcal{C})}{a+b} \tag{4}$$

Equation (4) is the update rule for off-diagonal elements in the kernel matrix. For diagonal elements, the expression is as follows:

$$K(\hat{\mathcal{A}}, \hat{\mathcal{A}}) = \frac{a^2K(\mathcal{A}, \mathcal{A}) + 2abK(\mathcal{A}, \mathcal{B}) + b^2K(\mathcal{B}, \mathcal{B})}{a^2 + 2ab + b^2} \tag{5}$$

Equations (4) and (5) are sufficient to update the kernel matrix after the merger of clusters $\mathcal{A}$ and $\mathcal{B}$. Each update requires constant time and updating the entire kernel matrix requires modifying $O(m)$ values in the matrix, where m is the number of clusters. The kernel matrix update procedure is illustrated in Figure 2.

VALIDATION OF CLUSTERING METHODS

Validation techniques for clustering methods fall into two categories: internal validation measures the quality of clusters based only upon the data, whereas external validation measures the agreement between the derived clusters and some external gold standard (Jain and Dubes, 1988). To evaluate kernel hierarchical clustering, we use one internal validation method and one external validation method.

Internal validation

The internal validation method measures the 'learnability' of a given set of clusters. Much work has been done evaluating clustering of gene expression data (Sharan and Shamir, 2000; Yeung *et al*., 2000); however, most such metrics require that the clustering be a complete partition. We specifically wanted to avoid a metric that evaluates the entire clustering because only the most prominent clusters in a given expression set are typically of interest. Given the relatively noisy character of microarray expression data, the more subtle clusters are likely to be indistinguishable from the noise.

We propose a learnability approach to internal cluster validation that determines the fitness of a given cluster by measuring the difference between the structure of the cluster and a structural prior. Our prior assumes that clusters are spherical in shape. This assumption corresponds to the learning bias of the k-nearest neighbor classifier (k-NN). k-NN is a supervised learning algorithm that takes as input a collection of labeled, n-dimensional points. The algorithm assigns to unclassified points the label of the majority of its k-nearest neighbors. We can imagine a ball, or hypersphere, in the input space centered at the unlabeled point, with radius equal to the distance to the kth nearest neighbor. The unlabeled point is assigned the label of the majority of the labeled points within the hypersphere.

The figure of merit we suggest is a learnability score obtained by cross validation using k-NN. The score is related to the jackknife resampling idea that motivates the figures of merit described by Yeung *et al*. (2000). In that work, the jackknife is performed by iteratively leaving out a single feature in each vector of the data set being clustered, and then comparing the resulting clusterings. In this work, we use 2-fold cross validation to evaluate the quality of a cluster. Rather than holding out a single vector for testing, we withhold half of the data, chosen at random. This technique reduces dependence problems caused by overlap of the training sets in the jackknife technique. We perform five replications of 2-fold cross validation to overcome variance due to the random sampling (Dietterich, 1998; Alpaydin, 1999).

For a given hierarchical clustering tree, our approach considers each subtree as a candidate cluster. We evaluate each cluster individually and use the membership status of each input vector to produce a binary labeling of the data. The data and labels are randomly partitioned into two roughly equal-sized sets. Each set, in turn, is used as the labeled training set while the remaining set is used as unlabeled test data. k-NN is used to classify the test set. After this procedure, each point is categorized as a true positive (TP) if k-NN predicts that it belongs to the cluster and it actually does belong to the cluster, a false positive (FP) if k-NN places it in the cluster when it does not belong there, a true negative (TN) if k-NN correctly places it outside the cluster and a false negative (FN) if k-NN incorrectly places it outside the cluster. The entire process is repeated five times.

For each candidate class, the number of TPs, FPs, TNs and FNs can be combined into a performance metric. For this metric, we use the product of precision and recall (PPR). In terms of TP, FP, and FN counts, the precision is expressed as $\mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$, and recall is expressed as $\mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$. Precision reflects how well the classifier rejects members of the negative class. Recall reflects how well a classifier identifies the members of the positive class. Ideally, the classifier would do well at both of these, which motivates taking the product of the measures. As mentioned above, we do not expect every gene within a microarray data set to fall into a biologically meaningful cluster. Therefore, in order to compare two different hierarchical clustering trees, we compute and compare the scores of the top ten non-overlapping clusters from each tree.

Table 2. Data sets used in this study

Data	Source	Genes	Annot	Arrays
Yeast	Eisen *et al*., 1998	6070	2465	79
Yeast	SMD	6112	2404	441
Mouse brain	Sandberg *et al*., 2001	10 000	6437	24
Leukemia	Golub *et al*., 1999	7129	4963	72
Colon cancer	Alon *et al*., 2000	2000	N/A	62

Column two lists the total number of genes included on the arrays; column three lists the number of genes that are annotated in the MYGD (yeast) or the Gene Ontology (mouse brain and leukemia).

External validation

The second validation method compares the given tree to an external gold standard. In this method, the gold standard consists of a large collection of possibly overlapping clusters. The clusters are identified using prior knowledge of the data, although not every cluster must appear in the data. In the evaluation, we aim to identify subtrees within the hierarchical clustering tree that correspond to classes within the gold standard. This is accomplished by computing PPR scores for each subtree with respect to each gold standard class. For each score, the maximal score for each cluster is selected as the best match. As in the internal validation, we consider only the top ten non-overlapping clusters from each tree.

METHODS

Data

The experiments compare clustering performance across five sets of gene expression data, summarized in Table 2. The first set, from Eisen *et al*., consists of 79 yeast microarray experiments from varying conditions, including the diauxic shift (DeRisi *et al*., 1997), the mitotic cell division cycle (Spellman *et al*., 1998), sporulation (Chu *et al*., 1998), and temperature and reducing shocks. Each array contains 6070 genes, of which Eisen *et al.* identified 2465 for which the function is known a priori. The annotations for these genes are derived from the MIPS Yeast Genome Database (MYGD) (Mewes *et al*., 2000). All 6070 genes are used for internal validation, but only the 2465 known genes are used for external validation.

The second data set is an expanded version of the first. The set consists of 441 yeast experiments collected from the Stanford Microarray Database (genome-www5.stanford.edu/MicroArray/SMD). This data set contains 6112 genes, with missing values imputed using a k-nearest neighbor approach (Troyanskaya *et al.*, 2001). Annotations are again derived from MYGD, resulting in a set of 2404 annotated genes.

The remaining three data sets are from studies of mouse and human. For these data sets, gene classifications are extracted from the Gene Ontology Consortium (2000). The mouse data set (Sandberg *et al.*, 2000) consists of 24 Affymetrix arrays, corresponding to six brain regions in two mouse strains with two-fold replication of each experiment. The first human data set (Golub *et al.*, 1999) consists of data from 72 human patients, each exhibiting one of two different types of acute leukemia. The final data set (Alon *et al.*, 1999) consists of 62 Affymetrix array experiments performed on 40 colon cancer tumors and 20 colon tissue samples from normal patients. Only the expression levels from the 2000 genes with highest minimal expression level across samples are publicly available. All 2000 genes are used in the experiments reported here. For this data set, no external validation is performed because functional annotations are not readily available.

Implementation

The kernel hierarchical clustering algorithm was implemented in C++. The implementation was validated by comparing the output to that produced by Cluster (rana.lbl.gov/EisenSoftware.htm). The software is available upon request from the authors.

Kernel functions

In any kernelized algorithm, the choice of kernel function must be made a priori. Because the kernel function determines the space in which the algorithm will operate, this choice is critical. Unfortunately, little is known about how best to select a kernel function for a given data set and algorithm. Accordingly, we adopt the standard empirical approach of experimenting with several common kernel functions. First, we use the first-degree polynomial kernel (POLY-1), $K(X, Y) = X \cdot Y$. This kernel was chosen as a baseline, because it amounts to not appyling the kernel trick: for POLY-1, the input space and feature space are identical. The POLY-2 and POLY-3 kernels are $K(X, Y) = [(X \cdot Y) + 1]^2$ and $K(X, Y) = [(X \cdot Y) + 1]^3$, respectively. The feature spaces induced by these kernels include features corresponding to all pairs or triples of features in the input space. Finally, the radial basis function (RBF) kernel

$$K(X, Y) = \mathrm{e}^{-||X-Y||^2/(2\sigma^2)} \tag{6}$$

is chosen for its ability to learn complex mappings. The RBF kernel function can learn, e.g., checkerboard patterns and other complex decision surfaces (Cristianini and Shawe-Taylor, 2000). The width σ is set to 0.01 in the results reported here. In the online supplement, results from additional values of σ are reported.

Table 3. Direct comparison of trees computed using standard and kernel-based centroid-link hierarchical clustering

	POLY-2	POLY-3	RBF
Yeast-79	26.5	26.6	39.7
Yeast-SMD	30.3	30.2	39.2
Mouse	75.6	70.6	81.9
Leukemia	57.9	56.5	66.7
Colon	64.3	70.0	76.3

Each number in the table is the percentage of branches in the given kernel tree that do not appear in the corresponding standard tree.

Table 4. Comparison of internal validation results

Algorithm	Data set	POLY-1	POLY-2	POLY-3	RBF
Centroid	Yeast-79	0.796	0.796	0.800	0.258
	Yeast-SMD	0.708	0.709	0.742	0.093
	Mouse	0.618	0.608	0.650	0.514
	Leukemia	0.751	0.728	0.742	0.472
	Colon	0.865	0.855	0.826	0.854
Average	Yeast-79	0.811	0.798	0.814	0.256
	Yeast-SMD	0.781	0.774	0.782	0.087
	Mouse	0.654	0.604	0.661	0.571
	Leukemia	0.763	0.721	0.761	0.457
	Colon	0.880	0.878	0.880	0.862
Single	Yeast-79	0.769	0.769	0.769	0.258
	Yeast-SMD	0.639	0.635	0.639	0.093
	Mouse	0.356	0.314	0.356	0.340
	Leukemia	0.565	0.534	0.565	0.453
	Colon	0.853	0.853	0.853	0.849
Complete	Yeast-79	0.751	0.742	0.751	0.321
	Yeast-SMD	0.743	0.743	0.743	0.047
	Mouse	0.680	0.601	0.680	0.509
	Leukemia	0.692	0.668	0.692	0.411
	Colon	0.854	0.854	0.854	0.792

Eighty trees were generated, corresponding to four clustering algorithms, four kernels, and five data sets. For each tree, 10 subtrees were selected that maximize the product of precision and recall. In the table, each entry is the average PPR across these ten subtrees. Due to computational constraints, the PPR scores for the mouse data set are based on 3000 genes chosen at random from the 6437 annotated genes.

RESULTS

Our experiments suggest that, although kernel hierarchical clustering produces trees whose topology differs depending upon the choice of kernel function, these differences are not likely to provide useful biological insight.

We first verify that the hierarchical clustering algorithm produces significantly different trees, depending upon the kernel

Table 5. Comparison of three kernels across five MYGD classes in the 79-experiment yeast data set using centroid-link clustering

		POLY-1				POLY-2				POLY-3				RBF			
Class	Size	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN
TCA	17	0	14	3	2448	0	14	3	2448	1	13	4	2447	2008	0	17	440
Resp	30	1	17	13	2434	0	17	13	2435	1	17	13	2434	2269	0	30	166
Ribo	121	12	15	106	2332	14	14	107	2330	14	15	106	2330	5	28	93	2339
Prot	35	1	9	26	2429	3	7	28	2427	7	5	30	2423	2267	0	35	163
Hist	11	0	2	9	2454	0	2	9	2454	0	2	9	2454	0	4	7	2454

The number of TPs, FPs, TNs and FNs is shown for five MYGD classes that were identified by Eisen *et al*. For each kernel and class, the subtree was selected that maximizes the PPR. The classes are tricarboxylic acid cycle, respiration chain complex, ribosomal proteins, proteasome, and histones.

function. For each data set, we compare the binary tree produced by standard hierarchical clustering with corresponding trees produced using non-linear kernel functions. Any tree can be described as a set of branches, in which each branch divides the leaves into two partitions. Two hierarchical clustering trees derived from the same data can therefore be compared with one another by counting the number of branches by which they differ. For some kernels and some types of clustering, the kernelization has no effect. In particular, the POLY-3 kernel does not change the similarity ranking of genes; hence, the kernel and non-kernel versions of the single-link and complete-link algorithms yield identical results with this kernel (see the supplementary data). In contrast, for $\sigma = 0.01$, the RBF kernel uniformly yields quite different results for all four clustering algorithms. Other settings yield smaller differences. Table 3 summarizes the comparisons for centroid-link clustering, in terms of the percentage of branches that differ between pairs of trees. In all cases, >25% of the branches in the standard hierarchical clustering tree do not appear in the corresponding kernel versions. Overall, these results show that kernelizing the hierarchical clustering algorithm often produces quite different trees.

Table 4 shows the results of internal validation of the various clusterings using the k-nearest neighbor approach. Although the PPR scores vary across the different kernel functions, there is no clear trend. If kernel hierarchical clustering provided a benefit, then we would expect the highest PPR scores to appear in the POLY-2, POLY-3 and RBF columns. This is not always the case.

For external validation, we first return to the five gene expression classes that were identified by Eisen *et al.* as clustering well in the Yeast-79 data set. These classes were also the subject of SVM analysis in the paper by Brown *et al.* In Table 5, the numbers of true and false positives and negatives are provided for each of these classes and for each kernel function. The table lists results for centroid-link clustering; results from the other clustering algorithms are similar (see the supplementary data). Overall, the classification performance of the clustering algorithm does not change dramatically as the kernel function is changed, except using the RBF kernel. In most cases, using a polynomial kernel results in a change of only a few misclassifications. The RBF results show that this version of the uniformly algorithm fails to find accurate clusters. These results are strongly dependent upon the width σ in Equation (6). For this data set, a larger value of σ yields results that are much closer to the results from the non-kernelized algorithm (see supplementary results). In no case, however, does the kernelized algorithm offer much improvement over the standard algorithm.

Similarly, the top-scoring classes for each of the data sets do not show any clear trends with respect to the choice of the kernel function. Table 6 lists the average PPR scores for all four data sets, four linkage criteria and four kernel functions. Detailed listings of the individual class scores are provided

Table 6. Summary of external validation results

Algorithm	Data set	POLY-1	POLY-2	POLY-3	RBF
Centroid	Yeast-79	0.546	0.549	0.541	0.426
	Yeast-SMD	0.454	0.457	0.445	0.259
	Mouse	0.364	0.358	0.359	0.358
	Leukemia	0.575	0.575	0.575	0.55
Average	Yeast-79	0.549	0.552	0.540	0.423
	Yeast-SMD	0.523	0.527	0.520	0.259
	Mouse	0.367	0.367	0.367	0.363
	Leukemia	0.616	0.616	0.616	0.55
Single	Yeast-79	0.523	0.523	0.523	0.427
	Yeast-SMD	0.442	0.442	0.442	0.259
	Mouse	0.350	0.350	0.350	0.350
	Leukemia	0.575	0.575	0.575	0.55
Complete	Yeast-79	0.489	0.461	0.489	0.279
	Yeast-SMD	0.517	0.516	0.516	0.259
	Mouse	0.363	0.363	0.363	0.363
	Leukemia	0.614	0.614	0.614	0.564

Each entry in the table is the average PPR score across the top ten classes found in a given hierarchical clustering tree. Results are given for each kernel function and each clustering algorithm.

in the online supplement. Overall, the rankings are quite similar to one another, even for the average-link and centroid-link algorithms, which we have seen yield quite different trees. For example, among the four different top ten rankings for the centroid-link algorithm and Yeast-SMD set, three rankings (POLY-1, POLY-2 and POLY-3) are re-orderings of the same 10 classes and the PPR scores for the outlier (RBF) are relatively low. A similar pattern is observed for the other three data sets.

DISCUSSION

We have demonstrated that, for these data sets and kernel functions, using kernel hierarchical clustering does not provide better results than using the standard version of the algorithm. A negative result such as the one reported here is always difficult to generalize. We cannot prove that kernel hierarchical clustering is not a good idea in general, only that it is not apparent that the algorithm provides a benefit in these experiments.

There are three primary reasons to report this negative result in the scientific literature. The first is to caution researchers who may in the future consider performing experiments such as these. Hopefully, this report will prevent repetition of our efforts.

The second motivation for this report is to provide insight into the idea of kernelization. The SVM and related large-margin classifiers were formulated specifically to operate well in very high-dimensional feature spaces. They are, in a sense, designed to combat the curse of dimensionality. As such, the kernel trick, which usually increases the dimensionality of the feature space, is less disadvantageous when used in conjunction with such an algorithm. Other algorithms, however, should be kernelized more judiciously.

Finally, this paper may also provide avenues for future research. Other clustering algorithms, such as k-means, may benefit more from kernelization than does hierarchical clustering. Also, it may be the case that kernel hierarchical clustering will be useful in conjunction with feature selection techniques that prevent the feature space from becoming unwieldy. Alternatively, the feature space might be derived from a generative model (Jaakkola and Haussler, 1998), thereby providing a concise set of features that, by incorporating appropriate prior knowledge, are more informative than the original data set. In combination with a specialized clustering algorithm (Tsuda *et al*., submitted for publication), such an approach may provide significant benefit in the clustering of microarray gene expression data.

ACKNOWLEDGMENTS

This paper was funded by a Sloan Research Fellowship to W.S.N. and by National Science Foundation grants DBI-0078523 and ISI-0093302.

REFERENCES

Alon,U., Barkai,N., Notterman,D.A., Gish,K., Ybarra,S., Mack,D. and Levine,A.J. (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, **96**, 6745–6750.

Alpaydin,E. (1999) Combined 5 × 2 cv f test for comparing supervised classification learning algorithms. *Neural Comput.*, **11**, 1885–1892.

Brown,M.P.S., Grundy,W.N., Lin,D., Cristianini,N., Sugnet,C., Furey,T.S., Ares,J.M. and Haussler,D. (2000) Knowledge-based analysis of microarray gene expression data using support vector machines. *PNAS*, **97**, 262–267.

Chu,S., DeRisi,J., Eisen,M., Mulholland,J., Botstein,D., Brown,P. and Herskowitz,I. (1998) The transcriptional program of sporulation in budding yeast. *Science*, **282**, 699–705.

Cristianini,N. and Shawe-Taylor,J. (2000) *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge.

DeRisi,J., Iyer,V. and Brown,P. (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, **278**, 680–686.

Dietterich,T. (1998) Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, **10**, 1885–1923.

Duda,R.O. and Hart,P.E. (1973) *Pattern Classification and Scene Analysis*. Wiley, New York.

Eisen,M., Spellman,P., Brown,P. and Botstein,D. (1998) Cluster analysis and display of genome-wide expression patterns. *PNAS*, **95**, 14863–14868.

Gene Ontology Consortium (2000) Gene ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–9.

Golub,T.R., Slonim,D.K., Tamayo,P., Huard,C., Gaasenbeek,M., Mesirov,J.P., Coller,H., Loh,M.L., Downing,J.R., Caligiuri,M.A. *et al*. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Jaakkola,T. and Haussler,D. (1998) Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*. Morgan Kauffmann, San Mateo, CA.

Jain,A. and Dubes,R. (1988) *Algorithms for Clustering Data*. Prentice Hall, New Jersey.

Mewes,H.W., Frishman,D., Gruber,C., Geier,B., Haase,D., Kaps,A., Lemcke,K., Mannhaupt,G., Pfeiffer,F., Schüller,C. *et al*. (2000) MIPS: a database for genomes and protein sequences. *Nucleic Acids Res.*, **28**, 37–40.

Mika,S., Rätsch,G., Weston,J., Schölkopf,B. and Müller,K.-R. (1999) Fisher discriminant analysis with kernels. In *Proceedings of the IEEE Neural Networks for Signal Processing Workshop 1999*.

Sandberg,R., Yasuda,R., Pankratz,D., Carter,T., Rio,J.D., Wodicka,L., Mayford,M., Lockhart,D. and Barlow,C. (2000) Regional and strain-specific gene expression mapping in the adult mouse brain. *PNAS*, **97**, 11038–11043.

Schölkopf,B., Smola,A. and Müller,K.-R. (1997) Kernel principal component analysis. In *Proceedings ICANN97*. Springer Lecture Notes in Computer Science, p. 583.

Sharan,R. and Shamir,R. (2000) CLICK: A clustering algorithm for gene expression analysis. In *Proceedings of the 8th International*

Conference on Intelligent Systems for Molecular Biology. AAAI Press, pp. 307–316.

Slonim,D.K. (2002) From patterns to pathways: gene expression data analysis comes of age. *Nat. Genet.* (Suppl. 2), 502–508.

Spellman,P.T., Sherlock,G., Zhang,M.Q., Iyer,V.R., Anders,K., Eisen,M.B., Brown,P.O., Botstein,D. and Futcher,B. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Troyanskaya,O., Cantor,M., Sherlock,G., Brown,P., Hastie,T., Tibshirani,R., Botstein,D. and Altman,R. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.

Tsuda,K., Kawanabe,M. and Müller,K.-R. (2003) Clustering with the Fisher score. In *Advances in Neural Information Processing Systems*. Morgan Kauffman (to appear).

Vapnik,V.N. (1998)*Statistical Learning Theory*. Adaptive and learning systems for signal processing, communications, and control. Wiley, New York.

Yeung,K.Y., Haynor,D.R. and Ruzzo,W.L. (2000) Validating clustering for gene expression data. Technical Report UW-CSE-00-01-01, University of Washington.