Simultaneous Classification and Feature Clustering Using Discriminant Vector Quantization with Applications to Microarray Data Analysis

Jia Li
Statistics Department
Penn State Univ., PA 16802
jiali@stat.psu.edu

Hongyuan Zha
Computer Science Department
Penn State Univ., PA 16802
zha@cse.psu.edu

Abstract

In many applications of supervised learning, automatic feature clustering is often desirable for a better understanding of the interaction among the various features as well as the interplay between the features and the class labels. In addition, for high dimensional data sets, feature clustering has the potential for improvement in classification accuracy and reduction in computational complexity. In this paper, a method is developed for simultaneous classification and feature clustering by extending discriminant vector quantization (DVQ), a prototype classification method derived from the principle of minimum description length using source coding techniques. The method incorporates feature clustering with classification performed by fusing features in the same clusters. To illustrate its effectiveness, the method has been applied to microarray gene expression data for human lymphoma classification. It is demonstrated that incorporating feature clustering improves classification accuracy, and the clusters generated match well with biological meaningful gene expression signature groups.

1 Introduction

Supervised classification has wide applications in a variety of science and engineering fields including bioinformatics, communication, information retrieval, and data mining. Due to rapid advances in information technology, we have witnessed an explosive growth in the amount as well as the complexity of scientific and engineering data, raising new challenges for developing effective and efficient data analysis methods. We now very often have to deal with data of very high dimensions, exacerbating the curse of dimensionality. To make the situation even more complicated, we also face data with a substantial amount of missing values resulted from imperfection in various experimental settings. Furthermore, to gain deeper understanding of a phenomenon we are interested in, we are no longer simply satisfied with high performance classification. Understanding the interaction among features and the interplay between features and the class labels is becoming more likely an integral part of the research goal.

Feature selection and feature clustering have been proposed as means for handling data complexity. The overall goals of feature selection for supervised learning include 1) improving classification accuracy; 2) reducing computational complexity. Often times with a smaller set of features more sophisticated classification models can be applied without assuming, for example, independence among the features. While sharing the goals of feature selection, feature clustering also seeks to identify clusters of features that have certain application-dependent meanings, for example, clusters of semantically-coherent words in textual document classification, and clusters of genes providing expression profile signatures for different types of cancerous tissue types in microarray data analysis. In this paper, we present a classification method that simultaneously predicts class labels and clusters features. Our unified approach attempts to cluster features so that the clustering causes minimum adverse effects on classification accuracy. On the other hand, taking classification into consideration is of interest even from the mere perspective of feature clustering since the aim in conventional clustering to optimize a defined measure of the tightness of clusters often lacks practically meaningful support. We will show that Discriminant Vector Quantization (DVQ) [16], a prototype classification algorithm, provides a natural mechanism to combine classification and feature clustering.

1.1 Previous Research

We focus on the prototype methods [10] of supervised classification, which are closely related to vector

quantization (VQ), a technology developed mainly for the purpose of data compression. Algorithms aimed at simultaneous compression and classification have been developed by Oehler, Gray, Perlmutter, Olshen, and Li [21, 17] through Bayes Vector Quantization and Tishby, Pereira, Bialek, and Slonim [25] through the Information Bottleneck Method. Algorithms that are purely for classification and use VQ as a tool include k-means [6, 9], Learning Vector Quantization [13, 12], and most recently Discriminant Vector Quantization [16]. See [16] for detailed review of these algorithms. Many methods for feature selection have been proposed in machine learning and statistics [14, 18, 19]. In [3], features are clustered using a version of the K-L divergence of the class label distributions as similarity measures. In [26], a similar feature clustering method was proposed using the above-mentioned Information Bottleneck framework. In all these methods feature clustering is used as a preprocessing step and feature clusters are computed based on certain *ad hoc* criteria. Furthermore, the feature clusters, once selected, remain fixed during the design of the classifier. In contrast, we seek to integrate feature clustering into classification with feature clusters computed iteratively to minimize an overall objective function.

There is a rich resource of literature on clustering methods applied to microarray gene expression data. Early work has been done by Eisen, Spellman, Brown, and Botstein [8] and Ben-Dor, Shamir, and Yakhini [4]. For an extensive review, readers are referred to [5]. Tibshirani, Hastie, Narasimhan, and Chu [28] recently investigated classifying cancer types using gene expression data.

The rest of the paper is organized as follows: We describe briefly Discriminant Vector Quantization in Section 2 as presented in [16]. In Section 3, a feature fusion model is established for simultaneous classification and clustering. The design of the classifier based on this model is presented in Section 4. The Bayes classification rule using the estimated model is provided in Section 5. How to treat missing data is discussed in Section 6. Feature selection is discussed in Section 7. The application of the algorithm to microarray data is described in Section 8. We conclude in Section 9.

2 Discriminant Vector Quantization

Assume the training data set is $\mathcal{L} = \{(x_i, y_i); i = 1, 2, ..., n\}$, where x_i is the feature vector and y_i is the class label. Typically (x_i, y_i) are considered as independent samples of random variables (X, Y). X is a continuous random variable in the d-dimensional Euclidean space R^d; and Y is a discrete random variable with finitely many possible values. Let $\mathcal{Y} = \{\infty, \in, ..., \mathcal{M}\}$ without loss of generality. A prototype method represents the data by a set of points (quantized vectors), or prototypes. Complying to the terminology of data compression, we also refer to a prototype as a codeword. Normally, a class is assigned to each prototype by majority vote on the associated class distribution of the prototype; and a test feature vector is identified as the class of its closest prototype.

Assume that the feature vector X and the class label Y are described by K prototypes with centroids $\hat{\mu}_k,\ k = 1, 2, ..., K$. The *probability mass function* (pmf) of Y in each prototype k is $b_k(y)$. By the principle of minimum description length (MDL) [22, 23], the optimal model provides the shortest description. This principle is applied to the design of prototypes by DVQ.

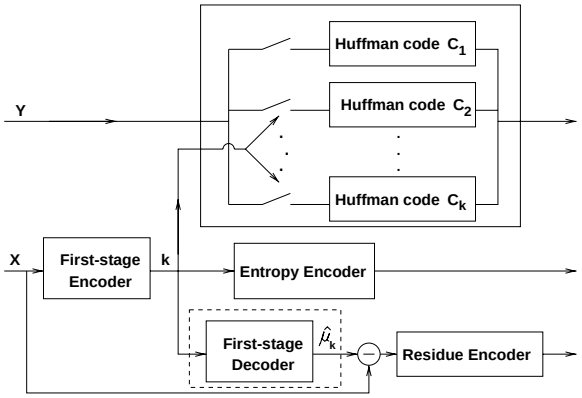

Figure 1: The encoding system for X and Y

DVQ employs an encoding system illustrated in Figure 1 to quantize X to a high precision and Y losslessly. In particular, the first stage VQ quantizes X to a prototype $\hat{\mu}_k$. The second stage VQ quantizes the residual between X and $\hat{\mu}_k$. To encode Y, a Huffman code is designed for each prototype. Denote the first stage encoder and decoder of X by α and β respectively, the residual encoder by $\tilde{\alpha}$, the entropy encoder of indices output by α by ζ, and the lossless encoder of Y by κ.

If we assume that the residual between X and its prototype is governed by the *probability density function* (pdf) $\phi(x)$ and the prior probability for X being in prototype k is a_k, the joint distribution of X and

Y, P_{XY}, is:

$$P_{XY}(x,y) = \sum_{k=1}^{K} a_k \phi(x - \hat{\mu}_k) b_k(y)$$
$$x \in R^d, y \in \{1, 2, ..., M\} . \quad (1)$$

Note that $\sum_{k=1}^{K} a_k = 1$. The pdf $\phi(x)$ is assumed to have zero mean and covariance matrix Σ. For simplicity, it is also assumed that $\phi(x)$ specifies a distribution with independent components of a vector. As a result, Σ is diagonal; and $\phi(x) = \prod_{j=1}^{d} \phi_j(x^{(j)})$, where $x^{(j)}$ is the jth dimension of vector x, a notation adopted throughout the paper.

In the spirit of the principle of MDL, the DVQ method designs the optimal encoding system and estimates parameters in Model (1) using the relation between those parameters and the optimal encoder. A recursive algorithm has been developed to improve the components, α, β, ζ, and κ, in the encoder iteratively. The distribution of residuals $\phi(x)$ can be estimated using a parametric model, e.g., Gaussian distribution. Although Gaussian distribution is restrictive with regard to fitting the residual data, it enjoys robustness due to low complexity.

The optimal classification rule (Bayes classification) that minimizes the average classification error rate based on Model (1) is

$$\begin{aligned} \hat{y} &= \max_{y \in \mathcal{Y}}{}^{-1} P\{Y = y \mid X = x\} \\ &= \max_{y \in \mathcal{Y}}{}^{-1} \sum_{k=1}^{K} a_k \phi(x - \hat{\mu}_k) b_k(y) . \end{aligned} \quad (2)$$

3 The Fusion Model for Classification and Feature Clustering

In this section, we extend Model (1) to incorporate feature clustering. The number of features is simply the dimension of X, denoted by d. We will use features and dimensions of X interchangeably in the sequel. Suppose the features are grouped into g clusters, $g \leq d$. Denote the mapping of a feature into its cluster by function $h(j)$, $j = 1, ..., d$. For features in cluster l, i.e., feature j's with $h(j) = l$, the values in dimension j's of every prototype $\hat{\mu}_k$ are assumed equal, that is, features in one cluster are not distinguished by the prototypes. We can thus "shrink" the d-dimensional vector $\hat{\mu}_k$ to g dimensions. From now on, $\hat{\mu}_k$ denotes a vector with g dimensions and its lth dimension $\hat{\mu}_k^{(l)}$ is the value in the kth prototype for all the features in cluster l. Extending Model (1), we have the following model for the joint distribution of X and Y reflecting feature clustering:

$$P_{XY}(x,y) = \sum_{k=1}^{K} a_k \prod_{j=1}^{d} \phi_j(x^{(j)} - \hat{\mu}_k^{(h(j))}) b_k(y) . \quad (3)$$

The above model is referred to as the fusion model as features in the same cluster are merged into one dimension in the prototypes. A model in the form of (3) tends to fit the data well when features in the same cluster are close to each other, for instance, in Euclidean distance. Consequently, fitting Model (3) implies simultaneous feature clustering. On the other hand, as a model for the joint distribution of the feature vector X and the class label Y, Model (3) leads to good classification if it fits the data well. The interplay of classification and feature clustering will become clearer when we present the algorithm to estimate the model.

In addition to the parameters shared by Model (1), the feature clustering function h in Model (3) needs to be estimated. The estimation of Model (3) based on optimal encoding is motivated by the principle of MDL. See [16] for detailed discussion. The two-stage encoding system in Figure 1 with modification is used to quantize X to a given high precision and to encode Y losslessly. The part in the dash line box in Figure 1 is modified, as shown in Figure 2. The design of the encoding system includes the feature clustering function $h(j)$ as well as the first stage encoder and decoder α, β, the entropy encoder ζ for the indices of prototypes, and κ, the classified entropy encoder of Y. Since β maps an index k output by the encoder α to a g-dimensional vector $\hat{\mu}_k$, to compare the prototype to a feature vector x_i, $\hat{\mu}_k$ is expanded to a d-dimensional vector with dimension j specified by $\hat{\mu}_k^{(h(j))}$. For brevity, with a bit abuse of notation, we denote this expanded vector by $h(\hat{\mu}_k)$. The two usages of h will be distinguishable from the context. The comparison between x_i and its prototype can therefore be expressed as the difference between x_i and $h(\beta(\alpha(x_i)))$.

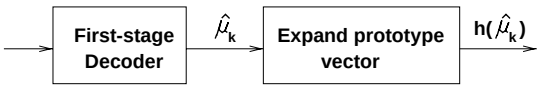

Figure 2: The modified part of the encoding system for X and Y with feature clustering

4 Design of the Classifier with Feature Clustering

4.1 Objective function

Notation used in the design of the encoding system presented in Figure 1 and 2 is introduced next. Assume $\alpha(x) = k$, if $x \in \mathcal{P}_k$, $k \in \{1, 2, ..., K\}$, $\bigcup_{k=1}^{K} \mathcal{P}_k = \mathcal{X}$, and $\beta(k) = \hat{\mu}_k$. Denote $P\{X \in \mathcal{P}_k\}$ by $q(k)$. Given α, the optimal entropy encoder for its output $\alpha(X)$ is the Huffman code [7] assigning a code length $\log \frac{1}{q(k)}$ to k. The fact $\log \frac{1}{q(k)}$ may not be an integer will be ignored. In the sequel, nats is used. Denote $P\{Y = m \mid X \in \mathcal{P}_k\}$ by $p(k, m)$. The optimal entropy encoder for Y given $X \in \mathcal{P}_k$ is a Huffman code assigning code length $\log 1/p(k, m)$ to class m. As shown in Figure 1, κ, the encoder for Y, contains multiple Huffman codes and selects one to encode Y according to $\alpha(X)$. To encode the residual of the first-stage quantizer, $X - h(\beta(\alpha(X)))$ is assumed to be governed by a multivariate Gaussian distribution with independent dimensions. The variance of each dimension, denoted by D_j, $j = 1, 2, ..., d$, is the mean squared error (MSE) in that dimension achieved by the first-stage quantizer $h \cdot \beta \cdot \alpha$. According to the rate distortion function of a univariate Gaussian distribution [7], on average, the number of nats needed to encode the jth dimension of the residual to a precision with MSE σ_j^2 is $\frac{1}{2} \log D_j/\sigma_j^2$. Since a Gaussian source represents the worst scenario for quantization given variance [15], $\frac{1}{2} \log D_j/\sigma_j^2$ is an upper bound on the average code length needed for the residual. To sum up, the average code length to encode $\{(X_i, Y_i)\}_{i=1}^n$ so that the jth dimension of X is quantized with distortion σ_j^2 and Y is encoded losslessly is,

$$nH(q) + \frac{n}{2} \sum_{j=1}^{d} \log \frac{D_j}{\sigma_j^2} + n \sum_{k=1}^{K} q(k) H(p(k)) \ ,$$

where $H(q)$ is the entropy of the pmf specified by $q(k)$, $k = 1, 2, ..., K$; and $H(p(k))$ is the entropy of the pmf specified by $p(k, m)$, $m = 1, 2, ..., M$, k fixed. The code length of the training sequence $\{(x_i, y_i)\}_{i=1}^n$ is

$$\sum_{i=1}^{n} \log \frac{1}{q(\alpha(x_i))} + \frac{n}{2} \sum_{j=1}^{d} \log D_j - \frac{n}{2} \sum_{j=1}^{d} \log \sigma_j^2 + \sum_{i=1}^{n} \log \frac{1}{p(\alpha(x_i), y_i)} \ , \qquad (4)$$

where $D_j = \frac{1}{n} \sum_{i=1}^{n} (x_i^{(j)} - \hat{\mu}_{\alpha(i)}^{(h(j))})^2$, $x^{(j)}$ denotes the jth dimension in the vector x; and $\hat{\mu}_{\alpha(i)}^{(h(j))}$ is the quantized value of $x^{(j)}$ by the first-stage encoder and decoder. For brevity, we use $\alpha(i)$ rather than $\alpha(x_i)$ to denote the prototype assigned to the ith data point (x_i, y_i).

The goal is to design α, β, κ, ζ, and h to minimize the following objective function

$$L(\alpha, \beta, \kappa, \zeta, h) = \sum_{i=1}^{n} \log \frac{1}{q(\alpha(x_i))} + \frac{n}{2} \sum_{j=1}^{d} \log D_j + \sum_{i=1}^{n} \log \frac{1}{p(\alpha(x_i), y_i)} \ , \qquad (5)$$

obtained from (4) by omitting the third term, which depends only on the preselected precision of X. The third term in (5), with expected value $n \sum_{k=1}^{K} q(k) H(p(k))$, reflects the penalty on misclassification. If $H(p(k))$ is low, the quantization cell $\mathcal{P}_k$ is "pure" in the sense that a certain class dominates. Hence, the probability of misclassification is low. The second term in (5) represents the cost resulted from two-way compression on both samples and features. The first term in (5) depends on the entropy of indices output by α. It represents to a certain extent the complexity of the quantized values. In summary, MDL leads to the design of a vector quantizer with an objective function incorporating compression distortion, the penalty of misclassification, and the complexity of quantized feature vectors.

4.2 Iterative Design

The five components α, β, ζ, κ, and h are updated recursively by alternating the optimization of each one with the other four fixed. The optimal entropy encoder ζ uses the Huffman code determined by the pmf $q(k)$, $k = 1, 2, ..., K$. The classified entropy encoder κ contains multiple Huffman codes determined by the pmfs $p(k, m)$, $m = 1, 2, ..., M$. For fixed β, ζ, κ, and h, α is not a nearest neighbor encoder, i.e., (x_i, y_i) is not necessarily mapped to the prototype closest to x_i in the Euclidean distance. On the other hand, given α, ζ, κ, and h, the optimal $\beta(k)$ is in general not the average of all the x_i's mapped to the kth prototype, a key difference from the basic DVQ due to the feature clustering function h. Next, we present the procedure to update α with the other components fixed. It is easy to see that when α is fixed, the optimal ζ and κ do not depend on the other components. The update of ζ and κ with fixed α is straightforward. As ζ and κ contain Huffman codes derived from pmfs $q(\cdot)$ and $p(\cdot, \cdot)$, the formula for updating $q(\cdot)$ and $p(\cdot, \cdot)$ are provided instead. The update of β and h will be introduced afterwards.

As the training data set is finite, searching for the optimal α is equivalent to determining n integer values

$\alpha(x_i)$, $i = 1, ..., n$, where n is the size of the training data set and $\alpha(x_i) \in \{1, ..., K\}$. Denote α at the beginning of iteration t by $\alpha^{(t)}$. Let $\hat{x}_i = h(\beta(\alpha^{(t)}(x_i)))$. The algorithm for updating $\alpha^{(t)}(x_i)$ to $\alpha^{(t+1)}(x_i)$ is as follows:

1. $1 \to i$;
2. $\frac{1}{n}\sum_{l=1}^{n}(x_l^{(j)} - \hat{x}_l^{(j)})^2 \to D_j$, $j = 1, 2, ..., d$.
3. $$\min_k{}^{-1}\left[\log\frac{1}{q(k)} + \log\frac{1}{p(k, y_i)} + \frac{n}{2}\sum_{j=1}^{d}\log(D_j + \frac{(x_i^{(j)} - \beta(k)^{(h(j))})^2 - (x_i^{(j)} - \hat{x}_i^{(j)})^2}{n})\right] \to \alpha^{(t+1)}(x_i)$$
4. $D_j + \frac{(x_i^{(j)} - \beta(\alpha^{(t+1)}(x_i))^{(h(j))})^2 - (x_i^{(j)} - \hat{x}_i^{(j)})^2}{n} \to D_j$, $j = 1, 2, ..., d$.
5. $h(\beta(\alpha^{(t+1)}(x_i))) \to \hat{x}_i$
6. $i + 1 \to i$. If $i > n$, stop; otherwise, go back to step 3.

After $\alpha^{(t+1)}$ is obtained, $q(\cdot)$ and $p(\cdot, \cdot)$ are updated by the following formula

$$q(k) = \sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k)/n\,, \tag{6}$$

$$p(k, m) = \frac{\sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k)I(y_i = m) + 1}{\sum_{i=1}^{n} I(\alpha^{(t+1)}(x_i) = k) + M}\,, \tag{7}$$

where $k = 1, 2, ..., K$ and $m = 1, 2, ..., M$. As usual $I(\cdot)$ is the indicator function that equals 1 if the argument is true and 0 otherwise. Note that to avoid classes with zero frequency, $p(k, m)$ is computed by a modified version of the normal frequency estimation.

When all the other four components are fixed, the minimization of the objective function in (5) associates with β only through the second term

$$\begin{aligned}&\frac{n}{2}\sum_{j=1}^{d}\log D_j\\ =\;& \frac{n}{2}\sum_{j=1}^{d}\log\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^{(j)} - \hat{\mu}_{\alpha(i)}^{(h(j))})^2\right)\,.\end{aligned} \tag{8}$$

The update of β is essentially searching for the optimal $\hat{\mu}_k^{(l)}$, $k = 1, ..., K$, $l = 1, ..., g$. For the convenience of illustration, we index features in cluster l by j_l and assume the number of features in cluster l is d_l, $\sum_{l=1}^{g} d_l = d$. Then Equation (8) can be written as

$$\begin{aligned}&\frac{n}{2}\sum_{j=1}^{d}\log D_j\\ =\;& \frac{n}{2}\sum_{l=1}^{g}\sum_{j_l}\log\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^{(j_l)} - \hat{\mu}_{\alpha(i)}^{(l)})^2\right)\,.\end{aligned} \tag{9}$$

Equation (9) demonstrates that $\hat{\mu}_k^{(l)}$, $k = 1, ..., K$, can be optimized separately for each l by minimizing

$$\sum_{j_l}\log\sum_{i=1}^{n}(x_i^{(j_l)} - \hat{\mu}_{\alpha(i)}^{(l)})^2\,.$$

Without loss of generality, let us derive $\hat{\mu}_k^{(1)}$, assuming $j_1 = 1, 2, ..., d_1$. Note that

$$\begin{aligned}&\sum_{j_l}\log(\sum_{i=1}^{n}(x_i^{(j_l)} - \hat{\mu}_{\alpha(i)}^{(l)})^2\\ =\;& \sum_{j_1=1}^{d_1}\log\sum_{s=1}^{K}\sum_{i=1}^{n}(x_i^{(j_1)} - \hat{\mu}_s^{(1)})^2 I(\alpha(i) = s)\\ \triangleq\;& u(\hat{\mu}_1^{(1)}, \hat{\mu}_2^{(1)}, ..., \hat{\mu}_K^{(1)})\,.\end{aligned} \tag{10}$$

To minimize the K variable function u, a descending algorithm is used to search for $\hat{\mu}_1^{(1)}$, $\hat{\mu}_2^{(1)}$, ..., $\hat{\mu}_2^{(K)}$ iteratively.

Suppose $\hat{\mu}_{s'}^{(1)}$ is to be solved to minimize function u with all the other variables fixed. Then a local minimum of u should satisfy

$$\begin{aligned}\frac{\partial u}{\partial\hat{\mu}_{s'}^{(1)}} &= \sum_{j_1=1}^{d_1}\frac{2\sum_{i=1}^{n}(\hat{\mu}_{s'}^{(1)} - x_i^{(j_1)})I(\alpha(i) = s')}{\sum_{s=1}^{K}\sum_{i=1}^{n}(x_i^{(j_1)} - \hat{\mu}_s^{(1)})^2 I(\alpha(i) = s)}\\ &= 0\,.\end{aligned} \tag{11}$$

Suppose the algorithm is at iteration t. Denote

$$T_{j_1,t} = \sum_{s=1}^{K}\sum_{i=1}^{n}(x_i^{(j_1)} - \hat{\mu}_s^{(1)})^2 I(\alpha(i) = s)\,,$$

where $\hat{\mu}_s^{(1)}$ are the current values at iteration t. Equation (11) can be approximately solved by

$$\hat{\mu}_{s'}^{(1)} = \frac{\sum_{j_1=1}^{d_1}\sum_{i=1}^{n} x_i^{(j_1)} I(\alpha(i) = s')/T_{j_1,t}}{\sum_{j_1=1}^{d_1}\sum_{i=1}^{n} I(\alpha(i) = s')/T_{j_1,t}}\,,$$

if modifying $\hat{\mu}_{s'}^{(1)}$ causes little change in the summation over all $\hat{\mu}_s^{(1)}$'s in $T_{j_1,t}$.

To sum up, the descending algorithm is as follows.

1. $0 \rightarrow t$
2. Set $\hat{\mu}_{s,t}^{(1)}$ to the values of $\hat{\mu}_s^{(1)}$ given by the decoder β before the current iteration of update.
3. For $j_1 = 1, ..., d_1$, set
$$T_{j_1,t} = \sum_{s=1}^{K} \sum_{i=1}^{n} (x_i^{(j_1)} - \hat{\mu}_{s,t}^{(1)})^2 I(\alpha(i) = s) .$$
4. $1 \rightarrow s'$
5. Set
$$w = \frac{\sum_{j_1=1}^{d_1} \sum_{i=1}^{n} x_i^{(j_1)} I(\alpha(i) = s')/T_{j_1,t}}{\sum_{j_1=1}^{d_1} \sum_{i=1}^{n} I(\alpha(i) = s')/T_{j_1,t}}$$
6. If
$$u(\hat{\mu}_{1,t}^{(1)}, ..., \hat{\mu}_{s'-1,t}^{(1)}, w, \hat{\mu}_{s'+1,t}^{(1)}, ..., \hat{\mu}_{K,t}^{(1)}) < u(\hat{\mu}_{1,t}^{(1)}, ..., \hat{\mu}_{s',t}^{(1)}, ..., \hat{\mu}_{K,t}^{(1)}) ,$$
set $\hat{\mu}_{s',t}^{(1)} = w$ and update
$$T_{j_1,t} = \sum_{s=1}^{K} \sum_{i=1}^{n} (x_i^{(j_1)} - \hat{\mu}_{s,t}^{(1)})^2 I(\alpha(i) = s) .$$
7. If $s' < K$, $s' + 1 \rightarrow s'$, go back to Step 5;
else {
If a stopping criterion is satisfied, stop;
else
$t + 1 \rightarrow t$, go back to Step 3. }

When all the other four components are fixed, the modification of h only affects the second term $\frac{n}{2} \sum_{j=1}^{d} \log D_j$ of the objective function in (5). Hence, the minimization of $L(\alpha, \beta, \kappa, \zeta, h)$ is equivalent to minimizing $\frac{n}{2} \sum_{j=1}^{d} \log D_j$. Furthermore, for a given dimension j', the value of $h(j')$ only affects $\log D_{j'}$ in the summation $\sum_{j=1}^{d} \log D_j$. It is thus sufficient to determine $h(j')$ by minimizing $D_{j'}$ separately for each j'. Recall $D_{j'} = \frac{1}{n} \sum_{i=1}^{n} (x_i^{(j')} - \hat{\mu}_{\alpha(i)}^{(h(j'))})^2$. Determined by the encoder α and decoder β respectively, the mapping $\alpha(i)$ and $\hat{\mu}_{\alpha(i)}^{(1)}$, $\hat{\mu}_{\alpha(i)}^{(2)}$, ..., $\hat{\mu}_{\alpha(i)}^{(g)}$ are all fixed in the update of h. The optimal $h(j)$ is thus

$$h(j') = \min_{l}{}^{-1} \sum_{i=1}^{n} (x_i^{(j')} - \hat{\mu}_{\alpha(i)}^{(l)})^2 . \quad (12)$$

Geometrically, $h(j)$ performs a nearest neighbor partitioning of the space of the features (dimensions). From the perspective of source coding, there are g codewords in the n-dimensional Euclidean space, the lth one specified by a row vector

$$(\hat{\mu}_{k(1)}^{(l)}, \hat{\mu}_{k(2)}^{(l)}, ..., \hat{\mu}_{k(n)}^{(l)}) .$$

The number of data points in the n-dimensional space is d, the jth one being a row vector

$$(x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)}) .$$

4.3 Initialization

To initialize α, β, κ, ζ, and h, we determine h, α, and β first. The initial κ and ζ are derived from Equation (6) and (7) after α is assigned. The initial $h(j)$, $j = 1, ..., d$ is derived from k-means clustering. Each feature j across all the samples can be viewed as a row vector of n dimensions $(x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)})$. K-means clustering in the n-dimensional vector space with d features yields an assignment of each feature j to one of the g clusters $h(j)$. For each cluster l, the feature closest to the centroid of the cluster is selected. These g features are used to design the initial α and β. For brevity of notation, assume without loss of generality that features 1, 2, ..., g are selected. K-means clustering is used again on the g-dimensional column vector $(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(g)})^t$, $i = 1, 2, ..., n$, to assign them to K prototypes. The initial $\alpha(x_i)$ is set to the prototype determined by k-means. The initial decoder β sets $\hat{\mu}_k$, $k = 1, ..., K$, to be the centroid of the kth prototype.

To choose the number of feature clusters g and the number of prototypes K, cross-validation [27] can be used if the goal is only to minimize classification error rate. We can also select g and K from the perspective of model selection. For instance, the Bayesian Information Criterion (BIC) [24] suggests that the optimal model ought to minimize a penalized log likelihood, particularly, $\sum_{i=1}^{n} \log P_{XY}(x_i, y_i) + \frac{N_p}{2} \log n$. N_p is the number of parameters in the model, increasing with g, K, and the sample dimension d. Other considerations for selecting g and K include computational complexity and practical issues depending on the specific problem at hand.

5 Bayes Classification

The performance of a classifier $\mathcal{K}$ is normally measured by the *Bayes risk*. Suppose the cost of labeling X as class $\hat{m}$ when the true class is m is $C_{m,\hat{m}}$. The Bayes risk is defined as

$$B(\mathcal{K}) = \sum_{m=1}^{M} \sum_{\hat{m}=1}^{M} C_{m,\hat{m}} P(\mathcal{K}(X) = \hat{m} \text{ and } Y = m) .$$

To minimize the Bayes risk, note that

$$\begin{aligned} B(\mathcal{K}) &= \sum_{m=1}^{M}\sum_{\hat{m}=1}^{M} C_{m,\hat{m}} P(\mathcal{K}(X)=\hat{m} \text{ and } Y=m) \\ &= E_X E_{Y|X} C_{Y,\mathcal{K}(X)} \\ &= E_X \sum_{m=1}^{M} P(Y=m \mid X) C_{m,\mathcal{K}(X)} \ . \end{aligned}$$

It is therefore sufficient to minimize the conditional risk $E_{Y|X=x}C_{Y,\mathcal{K}(x)}$ for each x. The optimal classification rule, referred to as the Bayes classifier, is given by

$$\mathcal{K}(x) = \min_{\hat{m}\in\mathcal{Y}}{}^{-1} \sum_{m=1}^{M} P(Y=m \mid X=x) C_{m,\hat{m}} \ .$$

Most frequently, $C_{m,\hat{m}} = 1$, if $m \neq \hat{m}$ and 0 otherwise. The resulting Bayes risk is the probability of misclassification. Correspondingly, the Bayes classifier is $\mathcal{K}(x) = \max_{\hat{m}}{}^{-1} P(Y=\hat{m} \mid X=x)$, i.e., the rule of majority vote. In this paper, Bayes risk is restricted to the probability of misclassification.

After the design of the optimal encoding system, the parameters in the fusion model (3) are estimated by $\hat{a}_k = q(k)$ and $\hat{b}_k(m) = p(k,m)$. The prototypes $\hat{\mu}_k$ and h are provided directly by the encoding system. Since we assume the distribution of the residuals $\phi(x) = \prod_{j=1}^{d} \phi_j(x^{(j)} - \hat{\mu}_k^{(h(j))})$ to be a multivariate Gaussian distribution with zero mean and independent dimensions, ϕ_j is uniquely determined by its variance, which is estimated by D_j. The classifier that minimizes classification error rate predicts the class of a feature vector x according to the Bayes rule:

$$\hat{y} = \max_{y\in\mathcal{Y}}{}^{-1} \sum_{k=1}^{K} \hat{a}_k \prod_{j=1}^{d} \phi_j(x^{(j)} - \hat{\mu}_k^{(h(j))}) b_k(y) \ .$$

6 Missing Data

In practice, the issue of missing data often arises. For instance, the microarray data set analyzed in Section 8 has 5% missing data overall. The percentage of samples with missing features is as high as 91% and the percentage of features with values missing in certain samples is 74%. It is thus impractical to use only samples without missing data. The issue of missing data can be dealt with naturally by the mechanism of DVQ without altering the criteria for estimation in training and for prediction of classes in testing. Prototypes are designed by DVQ based on the principle of MDL. The goal in the design is therefore to transmit the training data by the shortest codelength. If a sample has missing features, it will still be transmitted. But at the receiving end, values in these dimensions are considered as useless information.

In training and testing, there are two types of computation involving a missing dimension of a sample: summation and multiplication. If a term in a summation (product) uses a missing feature, that term is set to zero (one). Certain normalization factors for various summations may need to be adjusted due to the reduced amount of data. Missing features only affect the term $\frac{n}{2}\sum_{j=1}^{d} \log D_j$, $D_j = \frac{1}{n}\sum_{i=1}^{n}(x_i^{(j)} - \hat{\mu}_{\alpha(i)}^{(h(j))})^2$, in the objective function in (5). In general, that term ought to be $\sum_{j=1}^{d} \frac{n_j}{2} \log D_j$, where

$$D_j = \frac{1}{n_j}\sum_{i=1}^{n}(x_i^{(j)} - \hat{\mu}_{\alpha(i)}^{(h(j))})^2 I(x_i^{(j)} \text{ is not missing}) \, ,$$

and n_j is the number of samples without missing values of feature j. The Bayes rule for classifying a feature vector x with missing dimensions using the estimated joint distribution $\hat{P}_{XY}$ is straightforward. Without loss of generality, assume that only the first d' dimensions, $d' < d$, are known. The optimal class is:

$$\begin{aligned} \hat{y} &= \max_{y\in\mathcal{Y}}{}^{-1} P\{Y=y \mid X^{(1)}=x^{(1)}, \\ &\qquad X^{(2)}=x^{(2)}, ..., X^{(d')}=x^{(d')}\} \\ &= \max_{y\in\mathcal{Y}}{}^{-1} P\{Y=y, X^{(1)}=x^{(1)}, \\ &\qquad X^{(2)}=x^{(2)}, ..., X^{(d')}=x^{(d')}\} \\ &= \max_{y\in\mathcal{Y}}{}^{-1} \sum_{k=1}^{K} \hat{a}_k \prod_{j=1}^{d'} \phi_j(x^{(j)} - \hat{\mu}_k^{(h(j))}) b_k(y) \ . \end{aligned}$$

7 Feature Selection from Feature Clusters

Based on the feature clusters designed by the extended DVQ, a straightforward and *ad hoc* way to select features is to choose one representative feature from each cluster, e.g., the feature closest to the centroid of the cluster. The encoding system does not provide the centroids of the feature clusters directly. However, they can be computed from α and h by averaging. For cluster l, $l = 1, 2, ..., g$, the centroid is a row vector of n dimensions, $(\bar{\mu}_1^{(l)}, \bar{\mu}_2^{(l)}, ..., \bar{\mu}_n^{(l)})$, where $\bar{\mu}_i^{(l)}$ can be computed by

$$\bar{\mu}_i^{(l)} = \frac{\sum_{j=1}^{d} x_i^{(j)} I(h(j)=l)}{\sum_{j=1}^{d} I(h(j)=l)} \ .$$

Experiments with the microarray data described in Section 8 show that classification based only on

selected features tends to be inferior to classification by fusing features in the same cluster, especially when the number of features selected is small.

A more constructive approach to feature selection could be to modify the fusion model in (3) and the corresponding objective function $\mathcal{L}$ in (5) to reflect the requirement of feature selection rather than clustering. We avoid detailed discussion since the focus is on feature clustering here.

8 Application to Microarray Data

DNA microarray is a rapidly developing technology that allows the gene expression in an organism to be examined on a genomic scale, simultaneously measuring the transcription levels of tens of thousands of genes. In one particular application in cancer research, gene-expression profiling based on DNA microarray technology is expected to revolutionize cancer diagnosis. Our classification experiments are carried out based on data collected using 128 lymphochip microarrays designed to monitor genes involved in normal and abnormal lymphocyte development. The particular cancer studied is diffuse large B-cell lymphoma (DLBCL), a disease that takes in a clinically and morphologically varied group of tumors affecting the lymph system and blood [1]. The data set contains about 1.8 million measurements of gene expressions on 96 normal and malignant lymphocyte samples and is available at the following web site `http://llmpp.nih.gov/lymphoma/`. Each sample in the data set contains expression levels of 4026 genes; and the 96 samples are divided into nine classes. We chose four classes for the classification experiments because the other classes contain too few samples. The four classes are: 1) 46 samples of DLBCL (diffuse large B-cell lymphoma), 2) 11 samples of CL (follicular lymphoma), 3) 10 samples of ABB (Activated Blood B), 4) 9 samples of CLL (chronic lymphocytic leukemia).

In the experiment, the 76 samples of 4 classes are divided into a training set and a testing set of an equal number of samples. Classification error rates stated below are based on the testing data unless specified. We have experimented with 10, 50, 200, and 400 feature clusters. The number of prototypes used in training is 10. The classification error rates for all the four numbers of clusters are 5.3%. Similar classification results are obtained when the number of prototypes is around 10. Using too few prototypes yields low classification performance because of over-simplified modeling. On the other hand, since the training data set contains only 38 samples, 10 prototypes are used to ensure sufficient number of samples in each prototype. Experiments show, however, using more than 10 prototypes may yield lower classification error rates. For instance, when 20 prototypes are used with 10 feature clusters, the error rate is 2.6% for the testing data and 0.0% for the training data.

8.1 A closer look at the gene clusters generated

We have also manually examined two gene clusters for a classification run using 10 gene clusters. Among the 4026 genes, many have unknown functions. Some of the genes with known functions are identified in [1] with brief description of their biological functions and possible interactions. In Table 1 we list the known genes in the two gene clusters examined and indicate which gene profile signature groups they belong to. The numbers in the parentheses indicate the number of genes in each gene cluster. The first cluster contains genes corresponding to the lympho-node gene expression signature and T-cell gene expression signature [1]. Comparing this gene cluster with the two signature groups, we notice that only four types of genes in the T-cell gene expression signature are not included in this cluster: PKC-θ, fyn, T-cell receptor β chain, and Caspase. PKC-θ, fyn, and Caspase appear together in another cluster; and T-cell receptor β chain in yet another cluster. We examined the gene expression profiles for T-cell receptor β, fyn, and Caspase. They are quite similar for samples in DLBCL and ABB, but for samples in CLL and CL, there are prominent differences in T-cell receptor β versus fyn and Caspase. At this point, we do not know the biological significance of the differences. The second gene cluster contains genes corresponding to the proliferation gene expression signature. Missing from the second gene cluster are only SOS-1 genes; and again their expression profiles are relatively different from those in the second gene cluster.

9 Conclusion

In this paper, a method for classification with simultaneous feature clustering is presented. Extended from the approach of Discriminant Vector Quantization, a statistical classification model with feature fusion is proposed to integrate the optimization of classification and feature clustering. This model also allows missing data to be handled naturally without altering the estimation criterion and the Bayes classification rule. Feature selection based on the clusters generated has been explored. Motivated by the principle of minimum description length, the model is estimated by optimizing a source coding system. An efficient recursive algorithm has been developed to design the coding system. The application of the method

Table 1: Known genes in two gene clusters

Cluster 1 (209)	Cluster 2 (250)
CD14	Cyclic A
CD105	BUB 1
CSF-1 receptor	Cyclic B
FGF-7	SOCS-1
MMP-9	Ki67
NK4	p55CDc
TIMP-3	PLK
SDF-1	aurora-related kinase 1
Cathepsin B	P-16
Fc ϵ receptor γ chain	Thymidine kinase
integrin	CDC
LAT	RAD54
CD2	Dihydrofolate reductase
CD3	
CD49	
IRF-1	

to microarray gene expression data has demonstrated high performance classification and promising results of feature clustering.

References

[1] A. Alizadeh *et al.*, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling" *Nature*, 403(6769): 503-11, 2000.

[2] Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, **41**, 164-171.

[3] L.D. Baker and A.K. McCallum, "Distributional clustering of words for text classification," *Proceedings of SIGIR*, 96–103, 1998.

[4] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *Journal of Computational Biology*, vol. 6, pp. 281-97, 1999.

[5] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving submatrix problem," *Proc. RECOMB*, pp. 49-57, April 2002.

[6] R. K. Blashfield and M. S. Aldenderfer, "The literature on cluster analysis," *Multivariate Behavioral Research*, vol. 13, pp. 271-295, 1978.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

[8] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. Natl. Acad. Sci. USA*, vol. 95, pp. 14863-8, 1998.

[9] J. A. Hartigan and M. A. Wong, "Algorithm AS136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100-108, 1979.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning—Data Mining, Inference, and Prediction*, Springer, 2001.

[11] A. K. Jain, M. N. Murty and P. J. Flynn, "Data clustering: a review", *ACM Computing Surveys*, Vol. 31, pp. 264–323, 1999.

[12] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1989.

[13] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with Neural Networks: benchmarking studies," *IEEE Int. Conf. Neural Networks*, pp. I-61-68, July 1988.

[14] D. Koller and M. Sahami, "Toward Optimal Feature Selection", *International Conference on Machine Learning*, pp. 284-292, 1996.

[15] A. Lapidoth, "On the role of mismatch in rate distortion theory," *IEEE Trans. Inform. Theory*, vol. 43, pp. 38-47, Jan. 1997.

[16] J. Li, "A source coding approach to classification by vector quantization and the principle of minimum description length," *Data Compression Conference*, Snowbird, Utah, April 2002.

[17] J. Li, R. M. Gray, and R. A. Olshen, "Joint image compression and classification with vector quantization and a two dimensional hidden Markov model," *Data Compression Conference (DCC)*, pp. 23-32, Snowbird, Utah, March 1999.

[18] H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery and Data Mining," Kluwer Academic, 1998.

[19] A.J. Miller, "Subset Selection in Regression," New York: Chapman-Hall, 1990.

[20] K. L. Oehler and R. M. Gray, "Combining image compression and classification using vector quantization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 461-473, May 1995.

[21] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler, "Bayes risk weighted vector quantization with posterior estimation for image compression and classification," *IEEE Trans. Image Processing*, vol. 5, no. 2, pp. 347-360, Feb. 1996.

[22] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465-471, 1978.

[23] J. Rissanen, "Stochastic complexity and modeling," *The Annals of Statistics*, vol. 14, no. 3, pp. 1080-1100, 1986.

[24] Schwarz, G. (1978) Estimating the dimension of a model. *Annals of Statistics*, **6**, 461-464.

[25] N. Slonim and N. Tishby, "Agglomerative Information Bottleneck," *Proc. NIPS-12*, pp. 208-215, MIT Press, 1999.

[26] N. Slonim and N. Tishby, "The Power of Word Clusters for Text Classification," *23rd European Colloquium on Information Retrieval Research* , pp. 96–103, 2001.

[27] M. Stone, "Cross-validation: a review," *Math. Operationforsch. Statist. Ser. Statist.*, no. 9, pp. 127-139, 1978.

[28] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proc. Natl. Acad. Sci. USA*, vol. 99, pp. 6567-6572, May 2002.

[29] N. Tishby, F. C. Pereira, and W. Bialek, "The Information Bottleneck Method," *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pp. 368-377, 1999.