Segmentation of Multivariate Mixed Data via Lossy Data Coding and Compression

Yi Ma, *Senior Member*, *IEEE*, Harm Derksen,
Wei Hong, *Member*, *IEEE*, and John Wright, *Student Member*, *IEEE*

Abstract—In this paper, based on ideas from lossy data coding and compression, we present a simple but effective technique for segmenting multivariate mixed data that are drawn from a mixture of Gaussian distributions, which are allowed to be almost degenerate. The goal is to find the optimal segmentation that minimizes the overall coding length of the segmented data, subject to a given distortion. By analyzing the coding length/rate of mixed data, we formally establish some strong connections of data segmentation to many fundamental concepts in lossy data compression and rate-distortion theory. We show that a deterministic segmentation is approximately the (asymptotically) optimal solution for compressing mixed data. We propose a very simple and effective algorithm that depends on a single parameter, the allowable distortion. At any given distortion, the algorithm automatically determines the corresponding number and dimension of the groups and does not involve any parameter estimation. Simulation results reveal intriguing phase-transition-like behaviors of the number of segments when changing the level of distortion or the amount of outliers. Finally, we demonstrate how this technique can be readily applied to segment real imagery and bioinformatic data.

Index Terms—Multivariate mixed data, data segmentation, data clustering, rate distortion, lossy coding, lossy compression, image segmentation, microarray data clustering.

1 Introduction

Data that arise from practical problems in such diverse fields as image/signal processing, pattern recognition, computer vision, and bioinformatics are often characterized by complicated multimodal multivariate distributions. Segmentation (or clustering) is widely recognized as an important step in representing, analyzing, interpreting, or compressing such mixed data.

Now, the intriguing questions are listed as follows: What does "segmentation" really mean and how do we define it mathematically? What should the proper criterion for segmentation be and what do the segmentation results depend on? How should we measure the "gain" or "loss" of the segmentation? Last, but not the least, why is segmentation the right thing to do? Answers to these questions, to some extent, have been complicated by the many approaches and solutions for segmenting or modeling various types of mixed data proposed in the literature (see [1], [2], and the references therein for a review).

A somewhat traditional way of defining segmentation is to first choose a simple class of models that each subset is supposed to fit. Some of the popular models are either probabilistic distributions (for example, Gaussian distributions) or geometric/algebraic sets (for example, linear subspaces). Then, the whole mixed data are assumed to be samples drawn from a mixture of such probabilistic distributions [3], [4] or geometric/algebraic sets [5]. The typical approach to segmenting the data then entails estimating the mixture of all the models and then assigning each data point to the model with the highest likelihood.[1]

This way, data segmentation is essentially identified with a (mixture) model estimation problem. Segmenting the data and estimating the model are therefore strongly coupled together. Various approaches to resolve the coupling have been proposed in the literature:

- Iterate between the data segmentation and model estimation. Representative methods include the K-means algorithm [6], [7], [8], [9] (or its variants [10], [11], [12]) and the Expectation-Maximization (EM) algorithm [13], [14], which is essentially a greedy descent algorithm, to find the maximum likelihood (ML) estimate of a mixture of probabilistic distributions [3], [4], [15].
- Resolve the coupling between data segmentation and model estimation by first estimating a mixture model that does not depend on the segmentation of the data and then decomposing the mixture into individual components. Representative methods include the Generalized Principal Component Analysis (GPCA), in which the mixture model is assumed to be an arrangement of subspaces [5].

A common assumption behind all these approaches is that a good estimate of the underlying mixture model(s) is

- *Y. Ma and J. Wright are with the Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, Coordinated Science Laboratory, 1308 West Main Street, Urbana, IL 61801-2307. E-mail: {yima, jnwright}@uiuc.edu.*
- *H. Derksen is with the Department of Mathematics, University of Michigan, 530 Church Street, Ann Arbor, MI 48109-1043. E-mail: hderksen@umich.edu.*
- *W. Hong is with the DSP Solutions Research and Development Center, Texas Instruments, PO Box 660199, MS/8649, Dallas, TX 75266-0199. E-mail: weihong@ti.com.*

Manuscript received 7 Apr. 2006; revised 15 Aug. 2006; accepted 21 Nov. 2006; published online 18 Jan. 2007.
Recommended for acceptance by J. Buhmann.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0275-0406. Digital Object Identifier no. 10.1109/TPAMI.2007.1085.

1. Notice that there is another spectrum of clustering algorithms such as the classical agglomerative methods [25] that follow a different set of principles and do not explicitly cast the segmentation problem as density estimation.

necessary for the segmentation of the data. In a sense, the correctness of the segmentation relies on how good the estimate is. For instance, the given data $W = (w_1, w_2, \dots, w_m)$ are commonly assumed to be drawn from a mixture of distributions $p(x|\theta, \pi) \doteq \sum_{j=1}^{k} \pi_j p_j(x|\theta_j)$. When trying to obtain the optimal estimate of the mixture model, one usually chooses any of the model estimation criteria, for example, the ML estimate

$$(\hat{\theta}, \hat{\pi})_{ML} = \arg\max_{\theta,\pi} \sum_{i=1}^{m} \log p(w_i|\theta, \pi), \qquad (1)$$

where θ is the parameter of a certain class of (mixture) distributions of interest. The EM algorithm [13] (or its variants [16]) is often used to optimize the likelihood function of such a mixture model. The ML criterion is equivalent to minimizing the negated log-likelihood $\sum_i - \log p(w_i|\theta, \pi)$, which is approximately the expected coding length $\mathrm{Length}(W|\hat{\theta}, \hat{\pi})$ required to store the data by using the optimal coding scheme for the distribution $p(x|\hat{\theta}, \hat{\pi})$ [17].

When the number of component models k is not given a priori, we must estimate it from the data—a difficult task that is further complicated when the data are corrupted by a significant amount of outliers. To some extent, almost all model selection criteria used to determine the number of component models are equivalent to minimizing the coding length needed to describe both the data and the model, that is, the minimum description length (MDL) criterion [4], [18], [19], [20]

$$(\hat{\theta}, \hat{\pi})_{MDL} = \arg\min_{\theta,\pi} \; L(W, \theta, \pi) = L(W|\theta, \pi) + L(\theta, \pi), \qquad (2)$$

where the parameters θ, π are assumed to have a certain distribution $p(\theta, \pi)$. In general, the length function $L(\cdot)$ is chosen according to the optimal Shannon coding [17]: $-\log p(W|\theta, \pi)$ for W and $-\log p(\theta, \pi)$ for θ, π. Incidentally, this objective function coincides with the maximum a posteriori (MAP) estimate, and the EM algorithm again becomes the method of choice [4].

However, ML and MDL only truly correspond to minimum coding lengths when the random variables to be encoded are discrete.[2] For (multivariate) real-valued data, a finite coding length can only be obtained if we encode the data and the model parameters, *subject to a certain distortion* $\varepsilon > 0$. To this end, Madiman et al. [21] have studied the properties of *lossy ML* (LML) and *lossy MDL* (LMDL) criteria

$$(\hat{\theta}, \hat{\pi})_{LML} = \arg\min_{\theta,\pi} R(\hat{p}(W), \theta, \pi, \varepsilon), \qquad (3)$$

$$(\hat{\theta}, \hat{\pi})_{LMDL} = \arg\min_{\theta,\pi} R(\hat{p}(W), \theta, \pi, \varepsilon) + L(\theta, \pi), \qquad (4)$$

where $\hat{p}(W)$ is the empirical estimate of the probabilistic distribution from the data W. In [21], it is shown that (to first order, asymptotically) minimizing the coding rate of the data subject to the distortion ε is equivalent to computing the LML or LMDL estimate, with desirable properties such as (strong) consistency as an estimator. In our context, the coding rate (subject to a distortion) provides a natural measure of the goodness of segmentation for real-valued mixed data. In fact, the goal of modeling and segmentation of mixed data should indeed be consistent with that of data coding/compression: If the data can be fitted with better models after segmentation, then the data should be represented or encoded more efficiently with respect to such models.

2. Or, for continuous random variables, in the limit as the quantization error goes to 0.

1.1 Contributions of This Paper

In this paper, we do not consider modeling and segmenting data that have arbitrary mixture distributions. We are only interested in data that consist of multiple Gaussian-like groups, which may have significantly different and anisotropic covariances. The covariances of the groups may even be nearly degenerate, in which case we essentially want to fit the data with *multiple subspaces, possibly of different dimensions*. In this context, vector quantization (VQ) can be viewed as the special case of fitting the data with zero-dimensional (affine) subspaces [10].

Our approach to segmenting such mixed data follows the spirit of LML and LMDL. Our goal is to *find the optimal segmentation of the mixed data, which results in the shortest coding length subject to a given distortion of the data.* Our method, however, offers the following improvements over existing methods:

1. All of the estimates discussed above (ML, MDL, LML, and LMDL) are optimal only in an *asymptotic* sense, that is, for an infinite sequence of independent and identically distributed (i.i.d.) samples from the class of distributions of interest. In practice, however, we can only deal with finite (and often small) sets of samples. Thus, we introduce a measure of the coding length for each group, which not only closely approximates the optimal rate-distortion function for a Gaussian source [17], but also gives a tight upper bound for any finite number of samples.
2. We will prove that, with this choice of coding length/rate, *deterministic* segmentation is approximately asymptotically optimal, suggesting that probabilistic segmentation will not significantly reduce the overall coding length. This provides a theoretical justification that segmentation not only is useful for pragmatic purposes, but also well approximates the optimal solution for compressing data that are a mixture of Gaussians or subspaces.
3. An explicit formula for the coding length/rate function[3] allows one to directly evaluate the goodness of the segmentation. The tightness of the formula for small data sets leads to an efficient[4] "bottom-up" algorithm that minimizes the overall coding length by repeatedly merging small subsets, starting from individual data points. As we will show with extensive simulations and experiments, this approach resolves the difficult model selection issue [4] in an effective way, especially when the number of groups is unknown, or there is a significant amount of outliers.

3. This is the case for Gaussian sources. In general, computing the rate-distortion function for an arbitrary distribution is a difficult problem although many numerical methods exist in the literature (see [22] and the references therein).

4. The complexity of the proposed algorithm is polynomial in both size and dimension of the data.

4. When the level of distortion (or the density of outliers) varies continuously, the number of groups typically exhibits a *phase-transition* behavior similar to that in statistical physics, with the "correct" segmentation corresponding to one of the stable phases. Our simulations show that the number of segments need not be a monotonic function of the distortion.[5]

1.2 Organization of This Paper

We provide a summary of the basic ideas and the resulting algorithm of our approach in Section 2. In Section 3, based on the ideas from the rate-distortion theory in information theory, we introduce a formula for the coding rate/length needed to encode a set of vectors subject to a given distortion. An alternative verification of the formula is given in Appendix A and Appendix B shows how the formula should be modified when the data is nonzero mean. In Section 4, we study properties of the overall coding rate/length of mixed data after being segmented into multiple groups. Extensive simulation and experimental results of the proposed algorithm on synthetic and real data are given in Section 5.

2 Basic Ideas and Algorithm

In this section, we give a self-contained summary of the main ideas and algorithm of this paper and leave the more detailed mathematical analysis and justification to Sections 3 and 4. (Readers who are interested only in the algorithm and experiments may bypass these two sections and skip from Sections 2-5 without any loss of continuity.)

2.1 Lossy Coding of Multivariate Data

A lossy coding scheme maps a set of vectors $V = (v_1, v_2, \dots, v_m) \in \mathbb{R}^{n\times m}$ to a sequence of binary bits such that the original vectors can be recovered up to an allowable distortion $\mathbb{E}[\|v_i - \hat{v}_i\|^2] \leq \varepsilon^2$. The length of the encoded sequence is denoted as the function $L(V): \mathbb{R}^{n\times m} \to \mathbb{Z}_+$.

In general, the coding scheme and the associated $L(\cdot)$ function can be chosen to be optimal for any family of distributions of interest. In the case where the data are i.i.d. samples from a zero-mean[6] multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$, the function $R = \frac{1}{2}\log_2\det(I + \frac{n}{\varepsilon^2}\Sigma)$ provides a good approximation to the optimal rate-distortion function [17].[7] As $\hat{\Sigma} = \frac{1}{m}VV^T$ is an estimate of the covariance Σ, the average number of bits needed per vector is

$$R(V) \doteq \frac{1}{2}\log_2\det\Big(I + \frac{n}{\varepsilon^2 m}VV^T\Big). \tag{5}$$

For readers who are less familiar with the rate-distortion theory, we will give an intuitive explanation of this formula in Section 3.

Representing the m vectors of V therefore requires $mR(V)$ bits. Since the optimal codebook is adaptive to the data V, we must also represent it with additional $nR(V)$ bits,[8] yielding an overall coding length of

$$L(V) \doteq (m+n)R(V) = \frac{m+n}{2}\log_2\det\Big(I + \frac{n}{\varepsilon^2 m}VV^T\Big). \tag{6}$$

We will study the properties of this function in Section 3. For purposes of segmentation, it suffices to note that, in addition to being (approximately) asymptotically optimal for Gaussian data, $L(V)$ also provides a tight bound on the number of bits needed to code a finite number of vectors when the underlying distribution is a degenerate or nondegenerate Gaussian (see Appendix A for the proof).

2.2 Segmentation via Data Compression

Given a set of samples $W = (w_1, w_2, \dots, w_m) \in \mathbb{R}^{n\times m}$, one can always view them as drawn from a single Gaussian source and code W subject to distortion ε^2 by using $L(W)$ bits. However, if the samples are drawn from a mixture of Gaussian distributions or subspaces, it may be more efficient to code W as the union of multiple (disjoint) groups $W = W_1 \cup W_2 \cup \dots \cup W_k$. If each group is coded separately, then the total number of bits needed is

$$L^s(W_1, W_2, \dots, W_k) \doteq \sum_{i=1}^{k} L(W_i) + |W_i|(-\log_2(|W_i|/m)), \tag{7}$$

where $|W_i|$ indicates the cardinality (that is, the number of vectors) of the group W_i. In the above expression, the term $\sum_{i=1}^{k}|W_i|(-\log_2(|W_i|/m))$ is the number of bits needed to code (losslessly) the membership of the m samples in the k groups (for example, by using the Huffman coding [17]).[9]

Then, given a fixed coding scheme with its associated coding length function $L(\cdot)$, an optimal segmentation is one that minimizes the segmented coding length $L^s(\cdot)$ over all possible partitions of W. Moreover, we will see that, due to the properties of the rate-distortion function (5) for Gaussian data, softening the objective function (7) by allowing probabilistic (or fuzzy) segmentation does *not* further reduce the (expected) overall coding length (see Theorem 3 in Section 4).

Notice that the above objective (7) is a function of the distortion ε. In principle, one may add a "penalty" term such as $mn\log\varepsilon$ to the overall coding length[10] L^s so as to determine the optimal distortion ε^*. The resulting objective $\min_\varepsilon L^s + mn\log\varepsilon$ will then correspond to an optimal coding length that only depends on the data. Nevertheless, very often, we leave ε as a free parameter to be set by the user. In practice, this allows the user to potentially obtain a hierarchical segmentation of the data at different scales of quantization.

5. A different phase transition has been noticed in vector quantization using deterministic annealing, where the number of clusters increases monotonically when the annealing temperature decreases [10].

6. For simplicity, in the main text, we will derive and present our main results with the zero-mean assumption. However, all the formulas, results, and algorithms can be readily extended to the nonzero-mean case, as shown in Appendix B.

7. Strictly speaking, the rate-distortion function for the Gaussian source $\mathcal{N}(0,\Sigma)$ is $R = \frac{1}{2}\log_2\det(\frac{n}{\varepsilon^2}\Sigma)$ when $\frac{\varepsilon^2}{n}$ is smaller than the smallest eigenvalue of Σ. Thus, the approximation is good only when the distortion ε is relatively small. However, when $\frac{\varepsilon^2}{n}$ is larger than some eigenvalues of Σ, the rate-distortion function becomes more complicated [17]. Nevertheless, the approximate formula $R = \frac{1}{2}\log_2\det(I + \frac{n}{\varepsilon^2}\Sigma)$ can be viewed as the rate distortion of the "regularized" source that works for all ranges of ε. Furthermore, as we will show in Appendix A, the same formula gives a tight upper bound of the coding rate for any finite number of samples.

8. This can be viewed as the cost of coding the n principal axes of the data covariance $\frac{1}{m}VV^T$. A more detailed explanation of $L(V)$ is given in Section 3.

9. Here, we assume that the ordering of the samples is random and entropy coding is the best we can do to code the membership. However, if the samples are ordered such that nearby samples more likely belong to the same group (e.g., in segmenting pixels of an image), the second term can and should be replaced by a tighter estimate.

10. This particular penalty term is justified by noticing that $mn\log\varepsilon$ is (within an additive constant) the number of bits required to code the residual $w - \hat{w}$ upto (very small) distortion $\delta \ll \varepsilon$.

We will thoroughly examine how the value of ε affects the final segmentation through experiments in Section 5.

2.3 Minimizing the Coding Length

Finding the global minimum of the overall coding length L^s over all partitions of the data set is a daunting combinatorial optimization problem, intractable for large data sets. Nevertheless, the coding length can be effectively minimized in the steepest descent fashion, as outlined in Algorithm 1. The minimization proceeds in a "bottom-up" fashion. Initially, every sample is treated as its own group. At each iteration, two groups S_1 and S_2 are chosen so that merging them results in the greatest decrease in the coding length. The algorithm terminates when the coding length cannot be further reduced by merging any pair of groups.[11] A simple implementation that maintains a table containing $L^s(S_i \cup S_j)$ for all i and j requires $O(m^3 + m^2 n^3)$ time, where m is the number of samples, and n is the dimension of the space.

Algorithm 1 (Pairwise Steepest Descent of Coding Length).

```
1: input: the data $W = (w_1, w_2, \dots, w_m) \in \mathbb{R}^{n \times m}$ and a
          distortion $\varepsilon^2 > 0$.
2: initialize $\mathcal{S} := \{S = \{w\} | w \in W\}$.
3: while $|\mathcal{S}| > 1$ do
4:     choose distinct sets $S_1, S_2 \in \mathcal{S}$ such that
       $L^s(S_1 \cup S_2) - L^s(S_1, S_2)$ is minimal.
5:     if $L^s(S_1 \cup S_2) - L^s(S_1, S_2) \geq 0$ then break;
6:     else $\mathcal{S} := (\mathcal{S} \setminus \{S_1, S_2\}) \cup \{S_1 \cup S_2\}$.
7: end
8: output: $\mathcal{S}$
```

Extensive simulations and experiments demonstrate that this algorithm is consistently and remarkably effective in segmenting data that are a mixture of Gaussians or subspaces (see Section 5). It tolerates significant amounts of outliers and automatically determines the corresponding number of groups at any given distortion. As a greedy descent scheme, the algorithm does not guarantee to always find the globally optimal segmentation for any given (W, ε).[12] Based on our experience, we found that the main factor affecting the global convergence of the algorithm seems to be the density of the samples relative to the distortion ε^2. In Section 5, we will give strong empirical evidence for the convergence of the algorithm over a wide range of ε.

Notice that the greedy merging process in Algorithm 1 is similar in spirit to classical agglomerative clustering methods, especially Ward's method [24]. However, whereas Ward's method assumes isotropic Gaussians, our coding-based approach is capable of segmenting Gaussians with arbitrary covariances, including nearly degenerate distributions. Classical agglomerative approaches have been shown to be inappropriate for such situations [25]. In this sense, the change in coding length provides a principled means of measuring the similarity among arbitrary Gaussians. Our approach also demonstrates significant robustness to uniform outliers, another situation in which linkage algorithms [2] fail.

11. In the supplementary material (which can be found at http://computer.org/tpami/archives.htm), we have included a video showing the convergence of this algorithm on data drawn from mixtures of subspaces in $\mathbb{R}^3$.

12. However, it may be possible to improve the convergence by using more complicated split-and-merge strategies [16]. In addition, due to Theorem 1 of Section 4, the globally (asymptotically) optimal segmentation can also be computed via concave optimization [23], at the cost of potentially exponential computation time.

3 Lossy Coding of Multivariate Data

In this section, we give a more detailed justification of the coding rate/length functions introduced in Section 2. In Section 4, we provide a more thorough analysis of the compression-based approach to data segmentation. (Readers who are less concerned with technical details may skip these two sections at first reading without much loss of continuity.)

If the given data $w_i \in \mathbb{R}^n$ are i.i.d. samples of a random vector w with a probabilistic distribution $p(w)$, then the optimal coding scheme and the optimal coding rate of such a random vector w have been well studied in *information theory* (see [17] and references therein). However, here, we are dealing with a finite set of vectors $W = (w_1, w_2, \dots, w_m)$. Such a data set can be viewed as a nonparametric distribution itself: Each vector w_i in W occurs with an equal probability $1/m$. The optimal coding scheme for the distribution $p(w)$ is no longer optimal for W and the formula for the coding length is no longer accurate. Nevertheless, some of the basic ideas of deriving the optimal coding rate can still be extended to the nonparametric setting. In this section, by borrowing ideas from the information theory, we derive a tight bound of the coding length/rate for the given data W. In Appendix A, we give an alternative derivation of the bound. Although both approaches essentially arrive at the same estimate, they both reveal that the derived coding length/rate function holds under different conditions:

1. The derivation in this section shows that, for a small ε, the formula for $R(W)$ gives a good approximation to the (asymptotically) optimal rate-distortion function of a Gaussian source.
2. The derivation in Appendix A shows that the same coding length/rate formula works for any finite set of vectors W that span a subspace.

3.1 The Rate-Distortion Function

For simplicity, we here assume that the given data are zero mean; that is, $\mu \doteq \frac{1}{m}\sum_i w_i = 0$ (refer to Appendix B for the case in which the mean is not zero). Let ε^2 be the squared error allowable for encoding every vector w_i. That is, if $\hat{w}_i$ is an approximation of w_i, then we allow $\mathbb{E}[\|w_i - \hat{w}_i\|^2] \leq \varepsilon^2$. In other words, on the average, the allowable squared error for each entry of w_i is ε^2/n.

The solution to coding the vectors in W, subject to the mean squared error ε^2, can be explained by *sphere packing*, which is normally adopted in information theory [17]. Here, we are allowed to perturb each vector $w_i \in W$ within a sphere of radius ε in $\mathbb{R}^n$. In other words, we are allowed to distort each entry of w_i with an (independent) random variable of variance ε^2/n. Without loss of generality, we may model the error as an independent additive Gaussian noise:

$$\hat{w}_i = w_i + z_i, \quad \text{with} \quad z_i \sim \mathcal{N}\left(0, \frac{\varepsilon^2}{n} I\right). \tag{8}$$

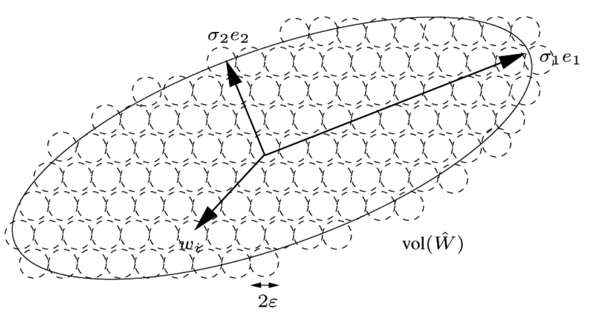


Fig. 1. Coding of a set of vectors in a region in $\mathbb{R}^n$ with an accuracy up to ε^2. To know the vector w_i, we only need to know the label of the corresponding sphere. e_1 and e_2 represent the singular vectors of the matrix $\hat{W}$, and σ_1 and σ_2 represent the singular values.

Then, the covariance matrix of the vectors $\hat{w}_i$ is

$$\hat{\Sigma} \doteq \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} \hat{w}_i \hat{w}_i^T\right] = \frac{\varepsilon^2}{n} I + \frac{1}{m} W W^T \in \mathbb{R}^{n\times n}. \qquad (9)$$

The volume of the region spanned by these vectors is proportional to the square root of the determinant of the covariance matrix

$$\mathrm{vol}(\hat{W}) \propto \sqrt{\det\left(\frac{\varepsilon^2}{n} I + \frac{1}{m} W W^T\right)}.$$

Similarly, the volume spanned by each random vector z_i is proportional to

$$\mathrm{vol}(z) \propto \sqrt{\det\left(\frac{\varepsilon^2}{n} I\right)}.$$

In order to encode each vector, we can partition the region spanned by all the vectors into nonoverlapping spheres of radius ε. When the volume of the region $\mathrm{vol}(\hat{W})$ is significantly larger than the volume of the sphere, the total number of spheres that we can pack into the region is approximately equal to

$$\#\text{ of spheres} = \mathrm{vol}(\hat{W})/\mathrm{vol}(z). \qquad (10)$$

Thus, to know each vector w_i with an accuracy up to ε^2, we only need to specify which sphere w_i it is in (see Fig. 1). If we use binary numbers to label all the spheres in the region of interest, the number of bits needed is

$$\begin{aligned} R(W) &\doteq \log_2(\#\text{ of spheres}) \\ &= \log_2(\mathrm{vol}(\hat{W})/\mathrm{vol}(z)) = \frac{1}{2}\log_2 \det\Big(I + \frac{n}{m\varepsilon^2} W W^T\Big), \end{aligned} \qquad (11)$$

where the last equality uses the fact that

$$\det(A)/\det(B) = \det(B^{-1}A).$$

If the samples w_i are drawn from a Gaussian source $\mathcal{N}(0,\Sigma)$, then $\frac{1}{m}WW^T$ converges to the covariance Σ of the Gaussian source. Thus, we have $R(W) \to \frac{1}{2}\log_2\det(I + \frac{n}{\varepsilon^2}\Sigma)$ as $m \to \infty$. When $\frac{\varepsilon^2}{n} \leq \lambda_{min}(\Sigma)$, the optimal rate distortion for a parallel i.i.d. $\mathcal{N}(0,\Sigma)$ source is $\frac{1}{2}\log_2\det(\frac{n}{\varepsilon^2}\Sigma)$, to which (11) provides a good approximation. In general, the optimal rate distortion is a complicated formula given by a reverse waterfilling on the eigenvalues of Σ (see Theorem 13.3.3 in [17]). The approximation (11) provides an upper bound that holds for all ε and is tight when ε is small relative to the eigenvalues of the covariance.

The formula for $R(W)$ can also be viewed as the rate distortion of the source W regularized by a noise of variance $\frac{\varepsilon^2}{n}$ as in (8). The covariance $\hat{\Sigma}$ of the perturbed vectors $\hat{w}_i$ always satisfies $\frac{\varepsilon^2}{n} \leq \lambda_{min}(\hat{\Sigma})$, allowing for a simple analytic expression for the rate distortion for all range of ε. This regularized rate distortion has the further advantage of agreeing with the bound for the coding length of finitely many vectors that span a subspace, derived in Appendix A. In addition, this formula resembles the channel capacity of a multiple-input multiple-output (MIMO) Gaussian channel (refer to Appendix C).

Notice that the formula for $R(W)$ is accurate only in the asymptotic sense, that is, when we are dealing with a large number of samples and the error ε is small (relative to the magnitude of the data W). We want to emphasize that the above derivation of the coding rate does not give an actual coding scheme. The construction of efficient coding schemes, which achieve the optimal rate-distortion bound, is itself a difficult problem (see, for example, [26] and references therein). However, for the purpose of measuring the quality of segmentation and compression, all that matters is that, *in principle*, a scheme attaining the optimal rate $R(W)$ exists.

3.2 The Coding Length Function

Given the coding rate $R(W)$, the total number of bits needed to encode the m vectors in W is

$$mR(W) = \frac{m}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2} W W^T\Big). \qquad (12)$$

Based on the communication point of view, $mR(W)$ bits are already sufficient, as both the transmitter and the receiver share the same codebook; that is, they both know the region spanned by W in $\mathbb{R}^n$. However, based on the data representation or the compression point of view, we need more bits to represent the codebook itself. This is equivalent to specifying all the principal axes of the region spanned by the data, that is, the singular values/vectors of W (see Fig. 1). As the number of principal axes is n, we need $nR(W)$ additional bits to encode them. Therefore, the total number of bits needed to encode the m vectors in $W \subset \mathbb{R}^n$ subject to the squared error ε^2 is[13]

$$L(W) \doteq (m+n)R(W) = \frac{m+n}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2} W W^T\Big). \qquad (13)$$

Appendix A provides an alternative derivation of the same coding length function $L(W)$ as an upper bound for a finite number of samples. If the data W have a nonzero mean, then we need more bits to encode the mean, too. (See in Appendix B how the coding length function should be properly modified in that case.)

13. Compared to the MDL criterion (2), if the term $mR(W)$ corresponds to the coding length for the data, the term $nR(W)$ then corresponds to the coding length for the model parameter θ.

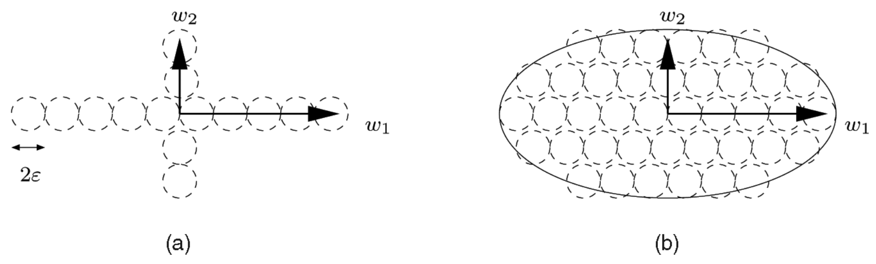


Fig. 2. The number of spheres (code words) of two different schemes for coding two orthogonal vectors. (a) Encoding the two vectors separately. (b) Encoding the two vectors together.

3.3 Properties of the Coding Length Function

3.3.1 Commutative Property

Since $WW^T \in \mathbb{R}^{n\times n}$ and $W^TW \in \mathbb{R}^{m\times m}$ have the same nonzero eigenvalues, the coding length function can also be expressed as

$$L(W) = \frac{m+n}{2}\log_2 \det\Big(I + \frac{n}{m\varepsilon^2}WW^T\Big) = \frac{m+n}{2}\log_2 \det\Big(I + \frac{n}{m\varepsilon^2}W^TW\Big).$$

Thus, if $n \ll m$, then the second expression will be less costly for computing the coding length. The matrix W^TW, which depends only on the inner products between pairs of data vectors, is known in the statistical learning literature as the *kernel matrix*. This property suggests that the ideas and the algorithm presented in Section 2 can be readily extended to segment data sets that have *nonlinear* structures by choosing a proper kernel function.

3.3.2 Invariant Property

Notice that, in the zero-mean case, the coding length function $L(W)$ is invariant under an orthogonal transformation of the data W. That is, for any orthogonal matrix $U \in O(n)$ or $V \in O(m)$, we have

$$L(UW) = L(W) = L(WV). \qquad (14)$$

In other words, the length function depends only on the singular values of W (or the eigenvalues of WW^T). This equality suggests that one may choose any orthonormal basis (for example, Fourier or wavelets) to represent and encode the data, and the number of bits needed should always be the same. This agrees with the fact that the chosen coding length (or rate) is optimal for a Gaussian source. However, if the data are non-Gaussian or nonlinear, a proper transformation can still be useful for compressing the data.[14] In this paper, we are essentially seeking a partition, rather than a transformation, of the non-Gaussian (or nonlinear) data set such that each subset is sufficiently Gaussian (or subspacelike) and, hence, cannot be compressed any further either by (orthogonal) transformation or by segmentation.

4 Coding Length of Segmented Data

Now, suppose we have partitioned the set of m vectors $W = (w_1, w_2, \ldots, w_m)$ into k nonoverlapping groups $W = W_1 \cup W_2 \cup \cdots \cup W_k$. Then, the total number of bits needed to encode the segmented data is

14. For a more thorough discussion on why some transformations (such as wavelets) are useful for data compression, the reader may refer to [27].

$$L^s(W_1, W_2, \ldots, W_k) = \sum_{i=1}^{k} L(W_i) + |W_i|(-\log_2(|W_i|/m)).$$

Here, the superscript "s" is used to indicate the coding length after segmentation.

4.1 Segmentation and Compression

To better understand under what conditions a set of data should or should not be segmented so that the overall coding length/rate becomes smaller, we here provide two representative examples. In the examples, we want to study whether a data set should be partitioned into two subsets of an equal number of vectors $W_1, W_2 \in \mathbb{R}^{n\times m}$. To simplify the analysis, we assume that $m \gg n$ so that we can ignore the asymptotically insignificant terms in the coding length/rate function.

Example 1 (Uncorrelated Subsets). Notice that, in general, we have

$$\begin{aligned} L(W_1) + L(W_2) &= \frac{m}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2}W_1W_1^T\Big) \\ &\quad + \frac{m}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2}W_2W_2^T\Big) \\ &\le \frac{2m}{2}\log_2\det\Big(I + \frac{n}{2m\varepsilon^2}\left(W_1W_1^T + W_2W_2^T\right)\Big) \\ &= L(W_1 \cup W_2), \end{aligned}$$

where the inequality is from the concavity of the function $\log_2\det(\cdot)$ (see Theorem 7.6.7 in [28]). Thus, if the difference $L(W_1 \cup W_2) - (L(W_1) + L(W_2))$ is large, then the overhead needed to encode the membership of the segmented data (here, one bit per vector) becomes insignificant. If we further assume that W_2 is a rotated version of W_1, that is, $W_2 = UW_1$ for some $U \in O(n)$, then one can show that the difference $L(W_1 \cup W_2) - (L(W_1) + L(W_2))$ is (approximately) maximized when W_2 becomes orthogonal to W_1. We call two groups W_1 and W_2 *uncorrelated* if $W_1^TW_2 = 0$. Thus, segmenting the data into uncorrelated groups typically reduces the overall coding length. From the viewpoint of sphere packing, Fig. 2 explains the reason.

Example 2 (Strongly Correlated Subsets). We say that two groups W_1 and W_2 are strongly correlated if they span the same subspace in $\mathbb{R}^n$. Or, somewhat equivalently, we may assume that W_1 and W_2 have approximately the same covariance $W_2W_2^T \approx W_1W_1^T$. Thus, we have

$$\begin{aligned} L(W_1) + L(W_2) &= \frac{m}{2}\log_2 \det\Big(I + \frac{n}{m\varepsilon^2} W_1 W_1^T\Big) \\ &\quad + \frac{m}{2}\log_2 \det\Big(I + \frac{n}{m\varepsilon^2} W_2 W_2^T\Big) \\ &\approx \frac{2m}{2}\log_2 \det\Big(I + \frac{n}{2m\varepsilon^2}\big(W_1 W_1^T + W_2 W_2^T\big)\Big) \\ &= L(W_1 \cup W_2). \end{aligned}$$

Since $L^s(W_1, W_2) = L(W_1) + L(W_2) + H(|W_1|, |W_2|)$, the overhead needed to encode the membership becomes significant and the segmented data require more bits than the unsegmented.

4.2 Optimality of Deterministic Segmentation

So far, we have only considered partitioning the data W into k nonoverlapping groups. That is, each vector is assigned to a group with a probability that is either 0 or 1. We call such a segmentation "deterministic." In this section, we examine an important question: *Is there a probabilistic segmentation of the data that can achieve an even lower coding rate?* That is, we consider a more general class of segmentations in which we assign each vector w_i to the group j according to a probability $\pi_{ij} \in [0, 1]$, with $\sum_{j=1}^k \pi_{ij} = 1$ for all $i = 1, 2, \ldots, m$.

To facilitate counting the expected coding length of such (probabilistically) segmented data, we introduce a matrix Π_j that collects the membership of the m vectors in group j:

$$\Pi_j \doteq \begin{bmatrix} \pi_{1j} & 0 & \cdots & 0 \\ 0 & \pi_{2j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \pi_{mj} \end{bmatrix} \in \mathbb{R}^{m\times m}. \qquad (15)$$

These matrices satisfy the constraint $\sum_{j=1}^k \Pi_j = I_{m\times m}, \Pi_j \succeq 0$.

Obviously, the jth group has an expected number of $\mathbf{tr}(\Pi_j)$ vectors, and the expected covariance is $\frac{1}{\mathbf{tr}(\Pi_j)} W\Pi_j W^T$. If viewed as a Gaussian source, the coding rate of the jth group is bounded by $R(W_j) \doteq \frac{1}{2}\log_2\det(I + \frac{n}{\mathbf{tr}(\Pi_j)\varepsilon^2} W\Pi_j W^T)$. If, for each vector w_i, we code it by using the coding scheme for the jth group with probability π_{ij}, then the expected total number of bits required to encode the data W according to the segmentation $\Pi = \{\Pi_j\}$ is bounded[15]

$$\begin{aligned} L^s(W,\Pi) \doteq \sum_{j=1}^k & \frac{\mathbf{tr}(\Pi_j) + n}{2} \log_2 \det\bigg(I + \frac{n}{\mathbf{tr}(\Pi_j)\varepsilon^2} W\Pi_j W^T\bigg) \\ & + \mathbf{tr}(\Pi_j)\bigg(-\log_2 \frac{\mathbf{tr}(\Pi_j)}{m}\bigg). \end{aligned} \qquad (16)$$

Similarly, the expected number of bits needed to encode each vector is bounded

$$\begin{aligned} R^s(W,\Pi) &\doteq \frac{1}{m} L^s(W,\Pi) \\ &= \sum_{j=1}^k \frac{\mathbf{tr}(\Pi_j)}{m}\bigg(R(W_j) - \log_2 \frac{\mathbf{tr}(\Pi_j)}{m}\bigg) + \frac{n}{m} R(W_j). \end{aligned} \qquad (17)$$

15. Strictly speaking, the formula is an upper bound for the expected coding length because $L^s(W,\Pi)$ is essentially a concave function of the group assignment Π (see the proof of Theorem 3). Hence, $L^s(W, \mathbb{E}[\Pi]) \geq \mathbb{E}[L^s(W,\Pi)]$ (using that $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$ for concave functions).

Thus, one may consider that the optimal segmentation Π^* is the global minimum of the expected overall coding length $L^s(W,\Pi)$ or, equivalently, the average coding rate $R^s(W,\Pi)$. To some extent, one can view the minimum value of $R^s(W,\Pi)$ as a good approximation to the actual entropy of the given data set W.[16]

Notice that $\frac{n}{m}R(W_j)$, the second term in the expression of $R^s(W,\Pi)$, is insignificant when the number of samples is large $m \gg n$. Nevertheless, this term, as well as the term that encodes the membership of the vectors, gives a *tight* bound on the coding length even for small sets of samples. This essentially allows us to find the optimal segmentation in a bottom-up manner by merging small subsets of samples, which is effectively harnessed by the greedy algorithm introduced in Section 2. That said, for the rest of this section, we examine more carefully the asymptotic properties of the coding length/rate function.

The first term in the expression of $R^s(W,\Pi)$ is the only part that matters asymptotically (that is, when the number of vectors in each group goes to infinity) and we denote it as

$$\begin{aligned} R^{s,\infty}(W,\Pi) \doteq \sum_{j=1}^k & \frac{\mathbf{tr}(\Pi_j)}{2m} \log_2 \det\bigg(I + \frac{n}{\varepsilon^2 \mathbf{tr}(\Pi_j)} W\Pi_j W^T\bigg) \\ & - \frac{\mathbf{tr}(\Pi_j)}{m}\log_2\bigg(\frac{\mathbf{tr}(\Pi_j)}{m}\bigg). \end{aligned}$$

Thus, the global minimum of $R^{s,\infty}(W,\Pi)$ determines the optimal segmentation when the sample size is large.

Theorem 3. *The asymptotic part $R^{s,\infty}(W,\Pi)$ of the rate-distortion function $R^s(W,\Pi)$ is a concave function of Π in the convex domain $\Omega \doteq \{\Pi : \sum_{j=1}^k \Pi_j = I, \Pi_j \succeq 0\}$.*

Proof. Let $\mathcal{S}$ be the set of all $m \times m$ nonnegative definite symmetric matrices. We will show that $R^{s,\infty}(W,\Pi)$ is concave as a function from $\mathcal{S}^k \to \mathbb{R}$ and also when it is restricted to the domain of interest $\Omega \subset \mathcal{S}^k$.

First, consider the second term of $R^{s,\infty}(W,\Pi)$. Notice that $\sum_{j=1}^k \mathbf{tr}(\Pi_j) = m$ is a constant. Thus, we only need to show the concavity of the function $g(P) \doteq -\mathbf{tr}(P)\log_2\mathbf{tr}(P)$ for $P \in \mathcal{S}$. The function $f(x) = -x\log_2 x$ is concave, and $g(P) = f(\mathbf{tr}(P))$. Therefore, for $\lambda \in [0, 1]$

$$\begin{aligned} & g(\lambda P_1 + (1-\lambda)P_2) = f(\lambda \mathbf{tr}(P_1) + (1-\lambda)\mathbf{tr}(P_2)) \\ & \geq \lambda f(\mathbf{tr}(P_1)) + (1-\lambda) f(\mathbf{tr}(P_2)) = \lambda g(P_1) + (1-\lambda) g(P_2). \end{aligned}$$

Thus, $g(P)$ is concave in P.

Now, consider the first term of $R^{s,\infty}(W,\Pi)$. Let

$$h(\Pi_j) \doteq \mathbf{tr}(\Pi_j)\log_2\det\bigg(I + \frac{n}{\varepsilon^2 \mathbf{tr}(\Pi_j)} W\Pi_j W^T\bigg).$$

It is well known in information theory that the function $q(P) \doteq \log_2\det(P)$ is concave for $P \in \mathcal{S}$ and $P \succ 0$ (see Theorem 7.6.7 in [28]). Now, define $r : \mathcal{S} \to \mathbb{R}$ to be

$$r(\Pi_j) \doteq \log_2\det\big(I + \alpha W\Pi_j W^T\big) = q\big(I + \alpha W\Pi_j W^T\big).$$

Since r is just the concave function q composed with an affine transformation $\Pi_j \mapsto I + \alpha W\Pi_j W^T$, r is concave (see Section 3.2.3 of [29]). Let $\psi : \mathcal{S} \times \mathbb{R}_+ \to \mathbb{R}$ as

16. Especially when the data W indeed consist of a mixture of subsets and each group is a typical set of samples from a (almost degenerate) Gaussian distribution.

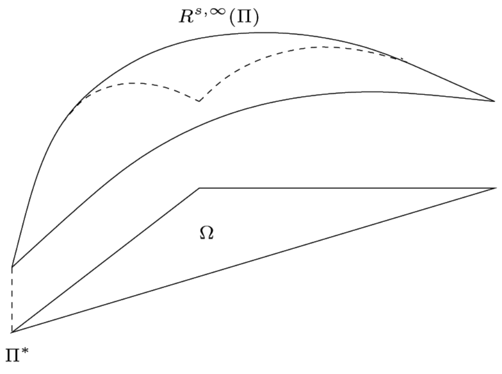


Fig. 3. The function $R^{s,\infty}(W,\Pi)$ is a concave function of Π over a convex domain Ω, which is, in fact, a polytope in the space $\mathbb{R}^{mk}$. The minimal coding length is achieved at a vertex Π^* of the polytope.

$$\psi(\Pi_j, t) \doteq t \cdot \log_2 \det\left(I + \frac{n}{\varepsilon^2 t} W \Pi_j W^T\right) = t \cdot r\left(\frac{1}{t}\Pi_j\right).$$

According to Theorem 3.2.6 in [29], ψ is concave. Notice that $H \doteq \{(\Pi_j, t) : t = \mathbf{tr}(\Pi_j)\}$ is a linear subspace in the product space of $\mathbb{R}$ and the space of all symmetric matrices. Therefore, $H \cap (\mathcal{S} \times \mathbb{R}_+)$ is a convex set, and the desired function $h(\Pi_j) = \psi(\Pi_j, \mathbf{tr}(\Pi_j))$ is just the restriction of ψ to this convex set. Thus, h is concave.

Since $R^{s,\infty}(W,\Pi)$ is a sum of the concave functions in Π_j, it is concave as a function from $\mathcal{S}^k$ to $\mathbb{R}$ and, so, is its restriction to the convex set Ω in $\mathcal{S}^k$. □

Since $R^{s,\infty}(W,\Pi)$ is concave, its global minimum Π^* is always reached at the boundary or, more precisely, at a vertex of the convex domain Ω, as shown in Fig. 3. At the vertex of Ω, the entries π_{ij} of Π^* are either 0s or 1s. It means that, even if we allow soft assignment of each point to the k groups according to any probabilistic distribution, the optimal solution with the minimal coding length can always be approximately achieved by assigning each point to one of the groups with a probability of 1. This is the reason why Algorithm 1 does not consider any probabilistic segmentation.

Another implication of the above theorem is that the problem of minimizing the coding length is essentially a concave optimization problem. Many effective concave optimization algorithms can be adopted to find the globally optimal segmentation, such as the simplex algorithm [23]. However, such generic concave optimization algorithms typically have high (potentially exponential) complexity. In Section 5, we will show with extensive simulations and experiments that the greedy algorithm proposed in Section 2 is already effective in minimizing the coding length.

Interestingly, in multiple-channel communications, the goal is instead to *maximize* the channel capacity, which has very much the same formula as the coding rate function. (See Appendix C for more detail.) The above theorem suggests that a higher channel capacity may be achieved inside the convex domain Ω, that is, by probabilistically assigning the transmitters into a certain number of groups. As the coding rate function is concave, the maximal channel capacity can be very easily computed via convex optimization [29].

5 Simulation and Experimental Results

In this section, we conduct simulations on a variety of challenging data sets to examine the effectiveness of the proposed coding length function as well the performance of the steepest descent algorithm. In the end, we will also demonstrate some experimental results of applying the algorithm to segment imagery and bioinformatic data.

5.1 Simulations

5.1.1 Segmentation of Linear Subspaces of Different Dimensions

We first demonstrate the ability of the algorithm to segment noisy samples drawn from a mixture of linear subspaces of different dimensions. For every d-dimensional subspace, $d \times 100$ samples are drawn uniformly from a ball of diameter 1 lying on the subspace. Each sample is corrupted with independent Gaussian noise of standard deviation $\varepsilon_0 = 0.04$. For Algorithm 1, we set $\varepsilon = \varepsilon_0$. We compare the results of Algorithm 1 with those of the EM algorithm for mixture of factor analyzers [30], followed by an ML classification step. We have modified the EM algorithm in [30] slightly to allow it to work for a mixture of factor analyzers with different dimensions. To avoid the model selection issue, which we postpone to Section 5.1.6, we provided the EM algorithm with the correct number and dimensions of the subspaces. Fig. 4 summarizes the comparison of results on several configurations tested.

In each case, the algorithm stops at the correct number of groups, and the dimensions of the segments W_i match those of the generating subspaces.[17] The correctness of the segmentation is further corroborated by the high percentage of points correctly classified (by comparing the segments with the a priori groups). For all five configurations, the average percentage of samples assigned to the correct group was at least 90 percent. The main cause of classification error is points that lie near the intersection of multiple subspaces. Due to noise, it may actually be more efficient to code such points according to the optimal coding scheme for one of the other subspaces. In all cases, Algorithm 1 dramatically outperforms EM (for a mixture of factor analyzers), despite requiring no knowledge of the subspace dimensions.

Since in practice ε_0 is not known, it is important to investigate the sensitivity of the results to the choice of ε. For each of the examples in Fig. 4, Table 1 gives the range of ε for which Algorithm 1 converges to the a priori number and dimension of subspaces. Notice that, for each of the configurations considered, there exists a significant range of ε for which the greedy algorithm converges.

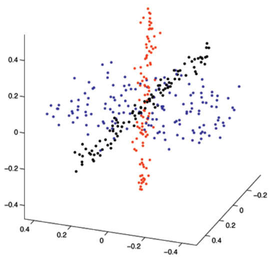
5.1.2 Global Convergence

Empirically, we find that Algorithm 1 does not suffer many of the difficulties with local minima that plague iterative clustering algorithms (for example, K-means) and parameter estimation algorithms (for example, EM). The convergence appears to depend mostly on the density of the samples relative to the distortion ε. For example, if the number of samples is fixed at $m = 1200$, and the data are drawn from three $\lceil \frac{n}{2} \rceil$-dimensional subspaces in $\mathbb{R}^n$, then the algorithm converges to the correct solution for $n = 2$ up to $n = 56$. Here, we choose $\varepsilon = \varepsilon_0 = 0.008$. Beyond $n = 56$, the algorithm fails to converge to the three a priori subspaces, as the samples have become too sparse. For $n > 56$, the computed

17. The dimension of each segment W_i is identified using principal component analysis (PCA) by thresholding the singular values of W_i with respect to ε.

Subspace dimensions	Identified dimensions	Classification (%) (Algorithm 1)	Classification (%) (EM)
$(2,1,1)$ in $\mathbb{R}^3$	$2,1,1$	96.62	39.33
$(2,2,1)$ in $\mathbb{R}^3$	$2,2,1$	90.00	68.98
$(4,2,2,1)$ in $\mathbb{R}^5$	$4,2,2,1$	98.53	43.36
$(6,3,1)$ in $\mathbb{R}^7$	$6,3,1$	99.77	66.16
$(7,5,2,1,1)$ in $\mathbb{R}^8$	$7,5,2,1,1$	98.04	42.29

(a)

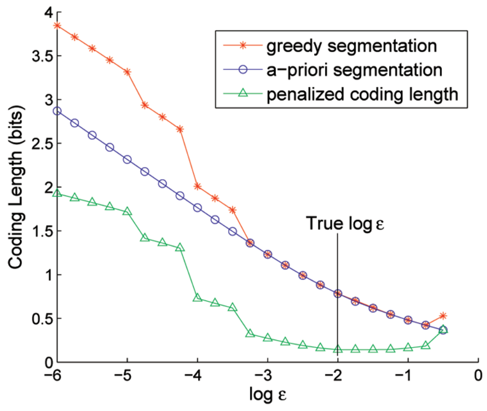


(b)

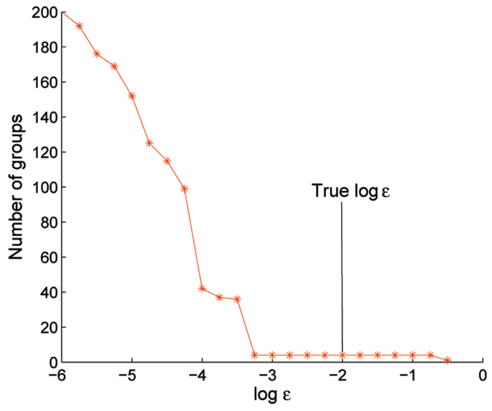
Fig. 4. (a) Simulation results for data drawn from mixtures of noisy linear subspaces. Classification percentages are averaged over 25 trials. Our algorithm correctly identifies the number and dimension of the subspaces in all 25 trials for all configurations. Far right column: results of using EM for mixture of factor analyzers with different dimensions [30] with random initialization. (b) The computed segmentation for (2, 1, 1) in $\mathbb{R}^3$ is displayed.

TABLE 1
The Size of the Range of $\log \varepsilon$ for Which the Algorithm 1 Converges to the Correct Number and Dimension of Groups for Each of the Arrangements Considered in Fig. 4

Subspace dimensions	$(2,1,1)$ in $\mathbb{R}^3$	$(2,2,1)$ in $\mathbb{R}^3$	$(4,2,2,1)$ in $\mathbb{R}^5$	$(6,3,1)$ in $\mathbb{R}^7$	$(7,5,2,1,1)$ in $\mathbb{R}^8$
$\log_{10}\varepsilon_{max} - \log_{10}\varepsilon_{min}$	2.5	1.75	2.0	2.0	.75



(a)



(b)

Fig. 5. (a) The coding length found by the greedy algorithm (the red curve) compared to the ground truth (the blue curve) for data drawn from four linear subspaces of dimension 20, 15, 15, 10 in $\mathbb{R}^{40}$. The green curve shows the penalized coding length $L^s + mn \log \varepsilon$. (b) The number of groups found by the greedy algorithm. It converges to the correct number, which is 4, when the distortion is relatively large.

segmentation gives a higher coding length than the a priori segmentation.

The same observation occurs for subspaces with different dimensions. For example, we randomly draw 800 noisy ($\varepsilon_0 = 0.14$) samples from four subspaces of dimension 20, 15, 15, 10 in $\mathbb{R}^{40}$. The results of the greedy algorithm at different distortions ε are shown in Fig. 5. As we see from the results, when the distortion ε is very small, the greedy algorithm does not necessarily converge to the optimal coding length. Nevertheless, the number of groups, which is 4, is still identified correctly by the algorithm when ε becomes relatively large.

As described in Section 2, ε can potentially be chosen automatically by minimizing $L^s + mn \log \varepsilon$, where the second term approximates (up to a constant) the number of bits needed to code the residual. The green curve in Fig. 5 shows the value of this penalized coding length. Notice that its minimum falls very near the true $\log \varepsilon$. We observe similar results for other simulated examples: the penalty term is generally effective in selecting a relevant ε.

5.1.3 Robustness to Outliers

We test the robustness of Algorithm 1 to outliers on the easily visualized example of two lines and a plane in $\mathbb{R}^3$. One hundred fifty-eight samples are drawn uniformly from a 2D disc of diameter 1. One hundred samples are drawn uniformly from each of the two line segments of length 1. The additive noise level is $\varepsilon_0 = 0.03$. The data set is contaminated with m_o outliers whose three coordinates are uniformly distributed on $[-0.5, 0.5]$.

As the number of outliers increases, the segmentation exhibits several distinct phases. For $m_o \leq 300$ (45.6 percent outliers), the algorithm always finds the correct segmentation. The outliers are merged into a single (3D) group. From $m_o = 400$ (52.8 percent outliers) up to $m_o = 1100$ (75.4 percent

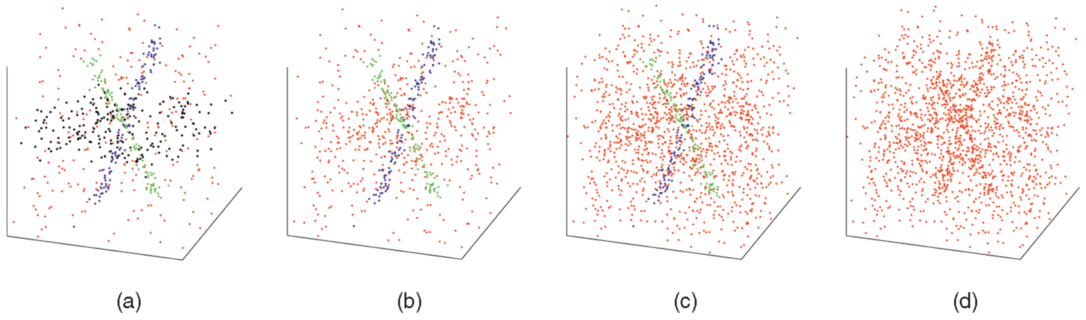


Fig. 6. Segmentation results for data drawn from three linear subspaces, corrupted with various numbers of outliers m_o. (a) $m_o = 300$ (45.6 percent outliers). (b) $m_o = 400$ (52.8 percent outliers). (c) $m_o = 1,100$ (75.4 percent outliers). (d) $m_o = 1,200$ (77.0 percent outliers).

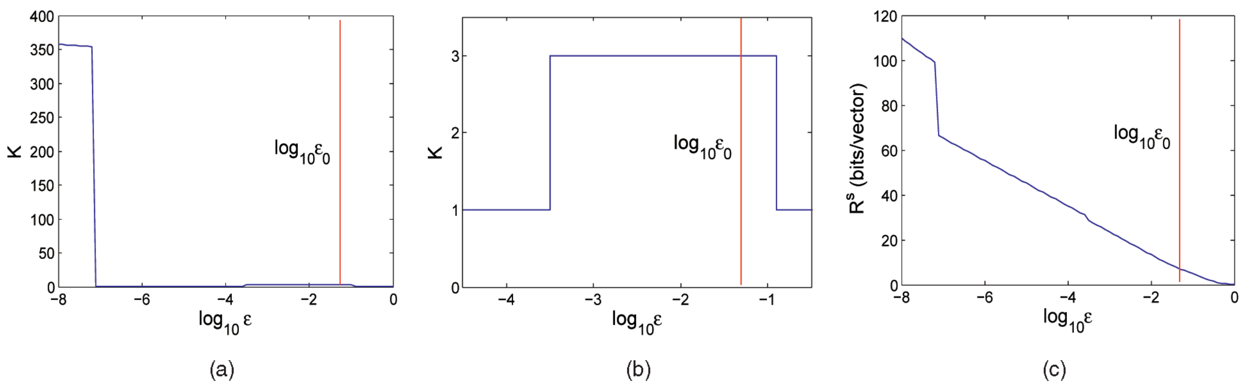


Fig. 7. The effect of varying ε, with $\varepsilon_0 = 0.05$. (a) The number of groups k versus $\log(\varepsilon)$. (b) Detail of k versus $\log(\varepsilon)$ around $\log(\varepsilon_0)$. (c) The coding rate (bits per vector) versus $\log(\varepsilon)$.

outliers), the two lines are correctly identified, but samples on the plane are merged with the outliers. For $m_o = 1,200$ (77.4 percent outliers) and above, all of the data samples are merged into one group, as the distribution of data has become essentially random in the ambient space. Fig. 6 shows the results for $m_o = 300, 400, 1,100, \text{and } 1,200$. Notice that the effect of adding the outliers (the lines and the plane) resembles the effect of ice being melted by warm water. This suggests a similarity between the artificial process of data clustering and the physical process of phase transition.

5.1.4 Number of Segments versus Distortion Level

Fig. 7 shows how the number of segments changes as ε varies. $m = 358$ points are drawn from two lines and a plane, as in the previous experiment and then perturbed with noise of standard deviation $\varepsilon_0 = 0.05$. Notice that the number of groups experiences distinct phases, with abrupt transitions around several critical values of ε. For sufficiently small ε, each data point is grouped by itself. However, as ε increases, the cost of coding the group membership begins to dominate, and all the points are grouped together in a single 3D subspace (the ambient space). Around the true noise level, ε_0, there is another stable phase, corresponding to the three a priori subspaces. Finally, as ε becomes large, the number of segments reverts to 1, as it becomes most efficient to represent the points using a single zero-dimensional subspace (the origin).

This behavior contrasts with the phase transition discussed in [10]. There, the number of segments increases monotonically throughout the simulated annealing process. Because our formulation allows the dimension of the segments to vary, the number of segments does not decrease monotonically with ε. Notice, however, that the phase corresponding to the "correct" (a priori) segmentation is stable over several orders of magnitude of the parameter ε. This is important since, in practice, the true noise level ε_0 is usually unknown.

Another interesting thing to notice is that the coding rate $R^s(W)$ in many regions is mostly a linear function of $-\log_{10}\varepsilon$: $R^s(W) \approx -\beta\log_{10}\varepsilon + \alpha$ for some constants $\alpha, \beta > 0$, which is a typical characteristic of the rate-distortion function of Gaussians.

For this data set, the algorithm takes about 10 seconds to run in Matlab on a 1.6-GHz personal computer.

5.1.5 Segmentation of Affine Subspaces

Appendix B shows how the coding length function should be properly modified in the case when the data are not zero-mean. Here, we show how the modified algorithm works for affine subspaces. We drew 358 samples from three linear subspaces in $\mathbb{R}^3$, and their centers are translated to $[2.1, 2.2, 2]^T$, $[2.4, 1.9, 2.1]^T$, and $[1.9, 2.5, 1.9]^T$.

Fig. 8 shows the segmentation results at different noise levels, with the distortion level chosen as $\varepsilon = \varepsilon_0$. For $10^{-7} < \varepsilon < 0.1$, the algorithm always identifies the correct number of subspaces with $\varepsilon = \varepsilon_0$. When $\varepsilon \leq 10^{-7}$, the density of the samples within the subspace becomes more important than the distortion orthogonal to the subspace, and the algorithm no longer converges with $\varepsilon = \varepsilon_0$. However, for such small distortion, there always exists a large stable phase (with respect to changing ε) corresponding to the correct number of subspaces $k = 3$. When $\varepsilon_0 > 0.1$, the algorithm starts to fail and merge the data samples into one or two groups.

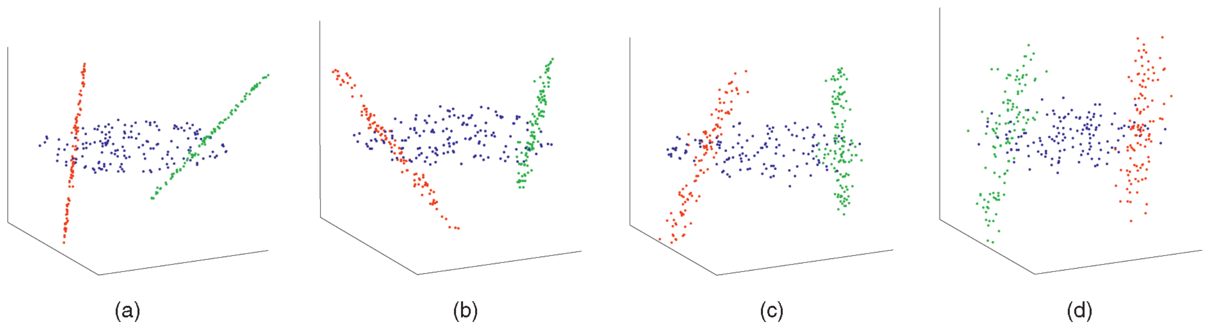


Fig. 8. The segmentation results for data drawn from three affine subspaces at different noise levels ε_0. The ε in the algorithm is chosen to be $\varepsilon = \varepsilon_0$. (a) $\varepsilon_0 = 0.01$. (b) $\varepsilon_0 = 0.03$. (c) $\varepsilon_0 = 0.05$. (d) $\varepsilon_0 = 0.08$.

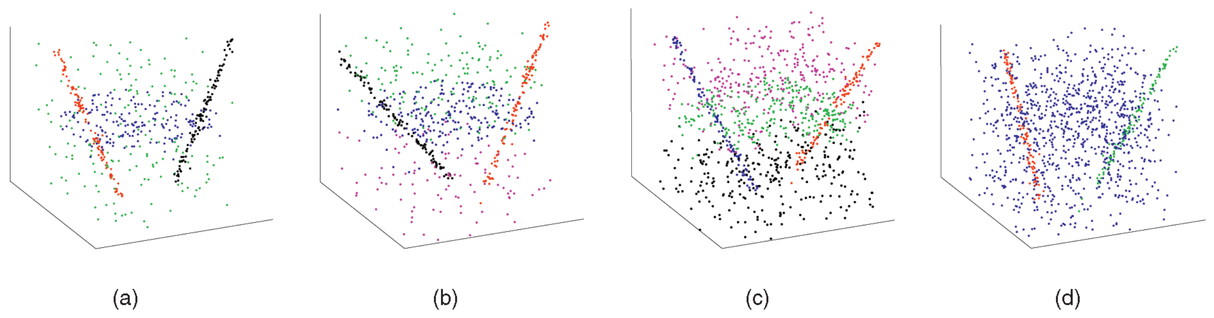


Fig. 9. The segmentation results for data drawn from three affine subspaces with different number of outliers m_o. The ε in the algorithm is $\varepsilon = \varepsilon_0 = 0.02$. (a) $m_o = 200$ (35.8 percent outliers). (b) $m_o = 300$ (45.6 percent outliers). (c) $m_o = 700$ (66.2 percent outliers). (d) $m_o = 800$ (69.1 percent outliers).

We now fix the Gaussian noise at $\varepsilon_0 = 0.02$ and add m_o outliers whose three coordinates are uniformly distributed in the range of [1.5, 2.5], which is the same as the range of the inliers. When the number of outliers is $\leq m_o = 200$ (35.8 percent outliers), the algorithm finds the correct segmentation, and all the outlying samples are segmented into one group. From $m_o = 300$ (45.6 percent outliers) to $m_o = 700$ (66.2 percent outliers), the algorithm still identifies the two lines and one plane. However, the outliers above and below the plane are clustered into two separate groups. For more than $m_o = 800$ (69.1 percent outliers), the algorithm identifies the two lines, but samples from the plane are merged with the outliers into one group. Fig. 9 shows the segmentation results $m_o = 200, 300, 700, 800$.

5.1.6 Model Selection for Affine Subspaces and Nonzero-Mean Gaussians

We compare the performance of Algorithm 1 to that of [4] and [31] on mixed data drawn from affine subspaces and nonzero-mean Gaussians. We test the algorithm's performance over multiple trials for three different types of data distribution. The first is three affine subspaces (two lines and one plane), with noise of standard deviation $\varepsilon_0 = 0.01$ and no outliers. Samples are drawn as in the previous examples. The means of the three groups are fixed (as in the previous examples), but the orientations of the two lines are chosen randomly. The second distribution tested is three affine subspaces (two planes and one line), with 158 points drawn from each plane and 100 from the line, again with $\varepsilon_0 = 0.01$. The orientations of one plane and of the line are chosen randomly. The final distribution tested is a mixture of $K = 3$ full-rank Gaussians in $\mathbb{R}^2$, with means [2, 0], [0, 0], and [0, 2] and covariance diag(2, 0.2; this is Fig. 3 of [4]). Nine hundred points are sampled (with uniform probability) from the three Gaussians.

For the two subspace examples, we run Algorithm 1 with $\varepsilon = \varepsilon_0 = 0.01$. For the third example, we set $\varepsilon = 0.2$. We repeat each trial 50 times. Fig. 10 shows a histogram of the number of groups arrived at by the three algorithms. For all algorithms, all of the segmentations with $K = 3$ are essentially correct (with the classification error being $<$ 4 percent). However, for degenerate or subspacelike data (Figs. 10a and 10b), Algorithm 1 was the most likely to converge to the a priori group number. For full-rank Gaussians (Fig. 10c), Algorithm 1 performs quite well but is outperformed by [4], which finds the correct segmentation in all 50 trials. The failures of Algorithm 1 occur because the greedy descent converges to a local minimum of the coding length rather than the global minimum.

Note that [4] was not explicitly designed for degenerate distributions, whereas [31] was not designed for full-rank distributions. Also, note that the samples in this experiment were drawn from a uniform distribution. The performance of each of the three algorithms improves when the generating distribution is indeed Gaussian. The main implication of the comparison is therefore that Algorithm 1 succeeds under a wide range of conditions and requires one to make less assumptions on the underlying data distribution.

5.2 Experiments on Real Data

In this section, we test the proposed segmentation method and algorithm on real imagery and bioinformatic data. Our goal here is to demonstrate that our method is capable of finding visually appealing structures in real data. However, we emphasize that it does not provide a complete solution to any of these practical problems. Such a solution usually

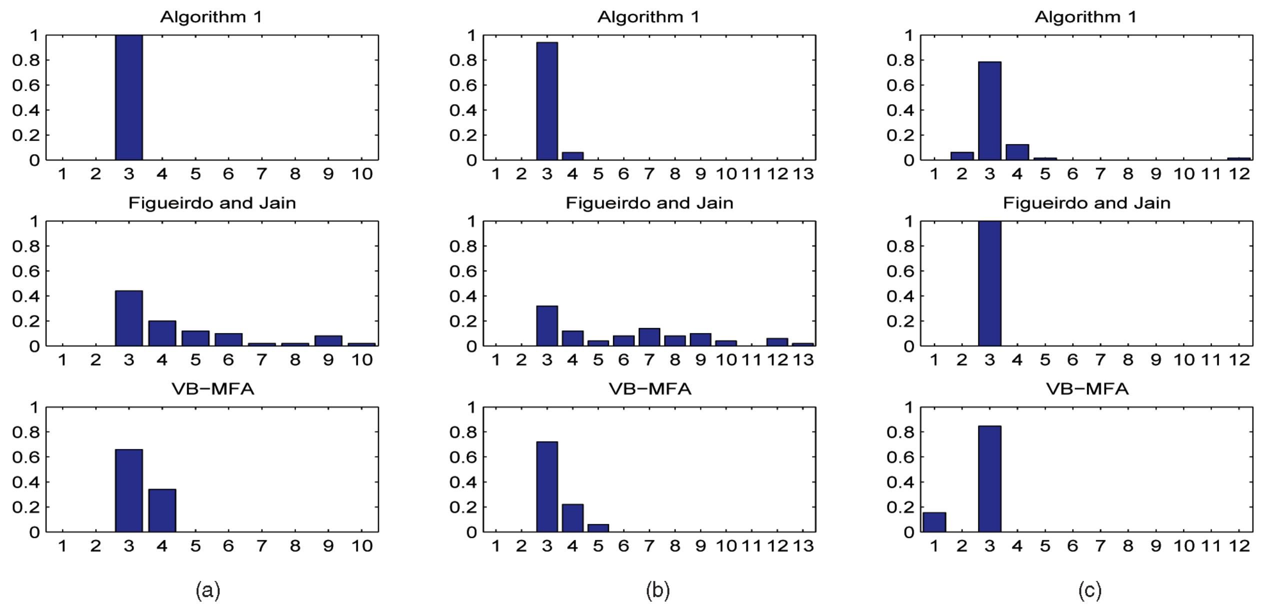


Fig. 10. Frequency of occurrence for various K in 50 trials. Top row: Algorithm 1. Middle row: [4]. Bottom row: [31]. (a) and (b) Results for randomly generated arrangements of affine subspaces. (c) Results for data sets generated from three full-rank Gaussians, as in [4]. For all cases, the correct number of groups is $K = 3$.

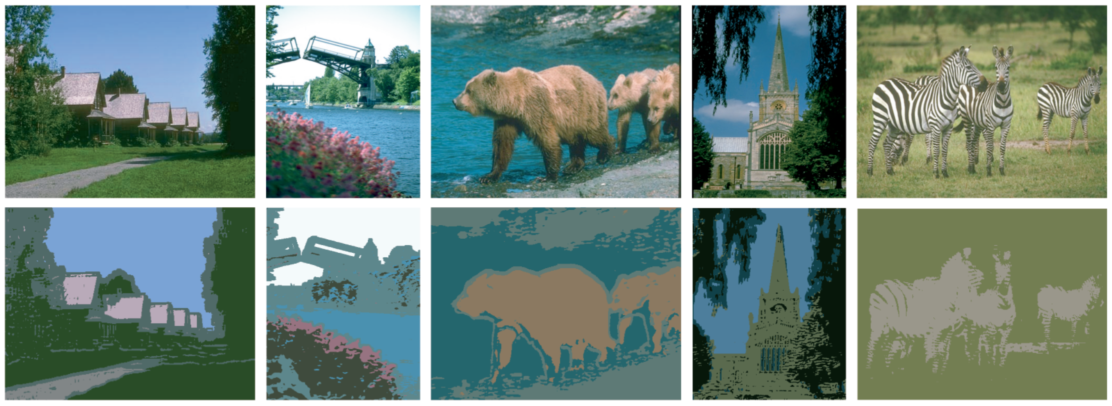


Fig. 11. Image segmentation (via the L formula for nonzero-mean Gaussian data), with $\varepsilon = 1$. Top row: original images. Bottom row: computed segmentations. Each segment is painted with its mean color.

entails a significant amount of domain-specific knowledge and engineering. Nevertheless, from these preliminary results with images and microarray data, we believe that the method presented in this paper provides a generic solution for segmenting mixed data, which is simple and effective enough to be easily customized for a broad range of practical problems.[18]

5.2.1 Image Segmentation

Fig. 11 shows the segmentation of several images from the Berkeley segmentation database via Algorithm 1 (using $L(\cdot)$ for nonzero-mean Gaussian data in Appendix B). The size of all the images is 480×320 pixels. We select an 8×8 window around each pixel to use as a feature vector[19] for segmentation. A random subset of 1,000 vectors are selected. PCA is applied to these vectors, and they are projected onto their first eight principal components. Subsampling and projection are necessary due to the sheer volume of data: For a 480×320 color image, we are dealing with 153,600 vectors in an $8 \times 8 \times 3 = 192$-dimensional space, beyond the computational power and memory of a personal computer. The subsampled and projected vectors are clustered using Algorithm 1, with $\varepsilon = 1$. The remaining vectors are then grouped with the nearest segment. Fig. 11 displays the results, without any further pre or postprocessing.

The segmentation can be further improved by first breaking the image into many small homogeneous regions via a superpixel step. We compute the superpixel oversegmentation by using a publicly available code [34]. We use its grouping to initialize the steepest descent procedure. To each pixel, we associate an 8×8 Gaussian-weighted window as a feature vector. Spatially adjacent groups are then repeatedly merged so as to achieve the greatest decrease in the coding length at each step. Fig. 12 shows some representative results from the Berkeley segmentation database. The results for the

18. At the time this paper is being prepared, we have also tested our algorithm on other mixed data such as speech and handwritten digits. The results are equally encouraging.

19. Raw pixel values provide a simple and intuitive feature for testing our approach on real data. More visually appealing segmentations might be obtained with more sophisticated features (e.g., filterbanks [32], [33]). We leave this to future work.

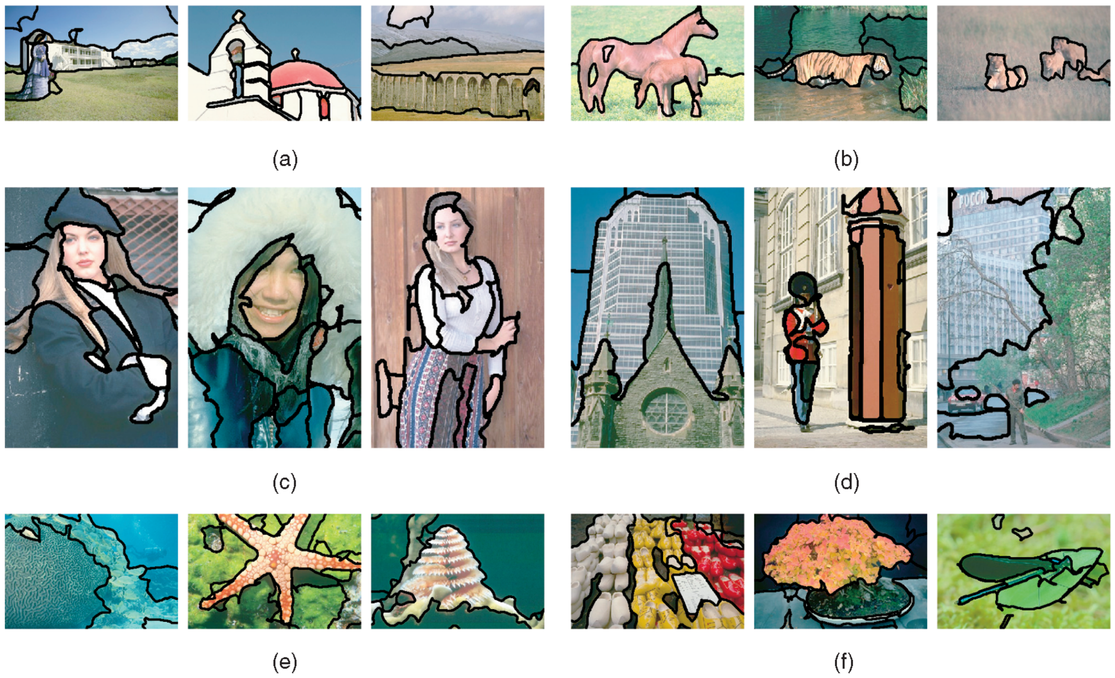


Fig. 12. Segmentation results for greedily merging adjacent segments to decrease the coding length. Here, the merging process is initialized via a superpixel oversegmentation. $\varepsilon = 0.02$ for all images. (a) Landscape. (b) Animals. (c) Portraits. (d) Urban. (e) Underwater. (f) Objects.

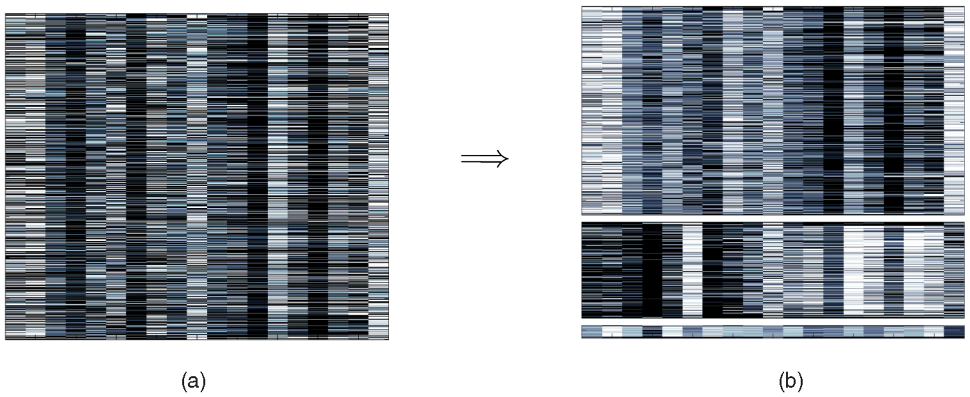


Fig. 13. Segmentation of microarray data. (a) Raw data. Each row represents the expression level of a single gene. (b) Three distinct clusters are found, visualized by reordering the rows.

entire database are available online at http://www.eecs.berkeley.edu/~yang/software/lossy_segmentation/. In quantitative terms, we find that our algorithm outperforms standard methods such as Normalized Cuts and Mean-Shift in terms of several common performance measures (for example, Variation of Information and Global Consistency Error). However, the performance in terms of the Boundary Distance measure is somewhat worse, perhaps due to the sensitivity of this measure to refinement (for more details, the reader is referred to [35]).

5.2.2 *Clustering of Microarray Data*

Fig. 13 shows the result of applying Algorithm 1 to gene expression data. The data set[20] consists of 13,872 vectors in $\mathbb{R}^{19}$, each of which describes the expression level of a single gene at different time points during an experiment on anthrax sporulation. A random subset of 600 vectors is visualized in Fig. 13a. Here, rows correspond to genes and columns to time points. We cluster these vectors without any preprocessing by using Algorithm 1, with $\varepsilon = 1$. The algorithm finds three distinct clusters, which are displayed in Fig. 13b, by reordering the rows.

Fig. 14 shows the clustering results from two additional gene expression data sets.[21] The first consists of 8,448 vectors in $\mathbb{R}^5$, describing the expression levels of yeast genes at five different time points during a heat shock experiment. Fig. 14a shows the expression levels for a randomly selected subset of 1,200 genes. We cluster these vectors by using Algorithm 1, with $\varepsilon = 0.1$. Our algorithm discovers a number of visually coherent clusters, shown in Fig. 14b. The second data set consists of 45,101 vectors in $\mathbb{R}^{10}$, each of which corresponds to the expression level of a single gene under varying

20. GDS930, available at http://www.ncbi.nlm.nih.gov/projects/geo.

21. GDS34 (left) and GDS1316 (right), also available at http://www.ncbi.nlm.nih.gov/projects/geo.

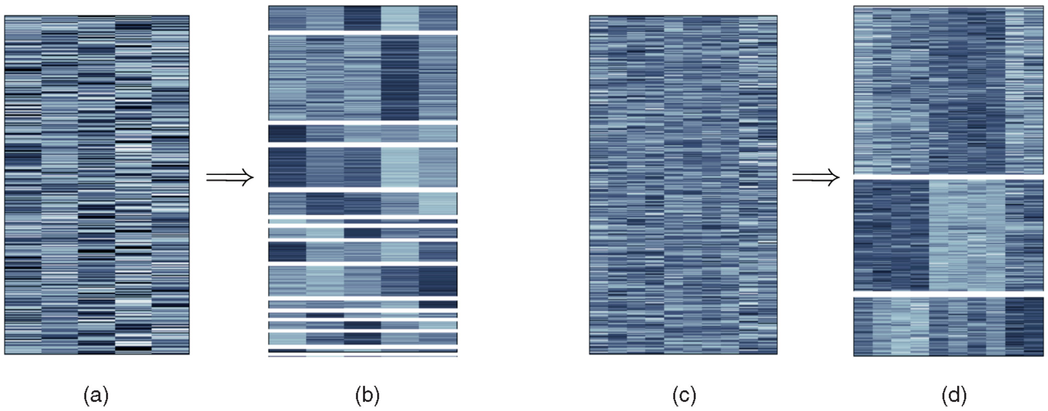


Fig. 14. Results from two microarray data sets. (a) Raw yeast data. (b) Segmentation, visualized by reordering rows. The algorithm discovers a number of distinct clusters of varying size. (c) Raw leukemia data. (d) Segmentation. Three clusters are found.

experimental conditions (this experiment investigated Down-syndrome-related leukemia). We run Algorithm 1, with $\varepsilon = 1$, on a subset of 800 vectors (shown in Fig. 14c). Three large distinct clusters emerge, shown in Fig. 14d, by reordering the rows of the data.

6 Conclusions and Discussions

In this paper, we have proposed a new approach to segment multivariate mixed data from a lossy data coding/compression viewpoint. Unlike most conventional model-based top-down approaches to segmenting the data, our work leads to a data-driven bottom-up approach to obtain the optimal segmentation. In addition, this new approach allows us to examine explicitly the effect of a varying distortion on the segmentation result. We find that the lossy-data-compression-based approach and the proposed greedy algorithm have the following attractive features:

1. The minimum coding length objective and the proposed greedy algorithm together deal with difficult issues such as outliers and model selection. The segmentation result is very stable with respect to distortion and noise and is very robust with respect to outliers.
2. The gain or loss of segmentation is measured by a *physically meaningful* quantity (binary bits) and the simulation results resemble the physical phenomenon of phase transition.
3. The greedy algorithm harnesses the tightness of the proposed coding length function for small sets of samples and takes a *bottom-up* approach that starts from merging the data one at a time. Thus, it needs no initialization and the optimal segmentation is obtained without knowing anything about the (underlying) subspace(s) or Gaussian model(s).
4. The greedy algorithm is *scalable*: Its complexity is polynomial in both the number of samples and the dimension of the data. The algorithm usually converges to the optimal solution as long as the distortion is reasonable with respect to the density of the samples.

Our analysis has shown connections of data segmentation with many fundamental concepts and results in information theory. The simulations and experiments have suggested potential connections with phase transition in statistical physics. From a theoretical standpoint, it would be highly desirable to obtain analytical conditions on the critical values of the distortion and the outlier density that can explain and predict the phase transition of the number of segments.

Moreover, we do not see any technical difficulty in extending this approach to supervised learning for purposes such as detection, classification, and recognition. It may also be extended to segment other types of structures such as non-Gaussian probabilistic distributions and nonlinear manifolds. As we mentioned earlier in the paper, there are many possible ways to improve the efficiency or convergence of the greedy algorithm or even develop new algorithms to minimize the coding length function. We will investigate such possibilities in the future.

Appendix A
Lossy Coding of Subspace-Like Data

In Section 3, we have shown that, in principle, one can construct a coding scheme for a given set of data $W \in \mathbb{R}^{n\times m}$ such that the average number of bits needed to encode each vector is bounded

$$R(W) = \frac{1}{2}\log_2 \det\Big(I + \frac{n}{m\epsilon^2}WW^T\Big) \tag{18}$$

if W is drawn from a multivariate Gaussian distribution of covariance $\Sigma = \frac{1}{m}WW^T$. However, we do not know in the nonparametric setting (that is, with finite number of samples) whether the above coding length is still of any good. In this appendix, we provide a constructive proof that $L(W) = (m+n)R(W)$ indeed gives a tight upper bound for the number of bits needed to encode W. One interesting feature of the construction is that the coding scheme apparently relies on coding the subspace spanned by the vectors (that is, the singular vectors) and the coordinates of the vectors with respect to the subspace. Thus, geometrically minimizing the coding length (via segmentation) is essentially equivalent to reducing the "dimension" (of each subset) of the data.

Consider the singular value decomposition (SVD) of the data matrix $W = U\Sigma V^T$. Let $B = (b_{ij}) = \Sigma V^T$. The column vectors of $U = (u_{ij})$ form a basis for the subspace spanned by the vectors in W, and the column vectors of B are the coordinates of the vectors with respect to this basis.

For coding purposes, we store the approximated matrices $U + \delta U$ and $B + \delta B$. The matrix W can be recovered as

$$W + \delta W \doteq (U + \delta U)(B + \delta B) = UB + \delta UB + U\delta B + \delta U\delta B. \tag{19}$$

Then, $\delta W \approx \delta UB + U\delta B$, as entries of $\delta U\delta B$ are negligible when ε is small (relative to the data W). The squared error introduced to the entries of W are

$$\sum_{i,j} \delta w_{ij}^2 = \mathbf{tr}\big(\delta W \delta W^T\big) \approx \mathbf{tr}\Big(U\delta B\delta B^T U^T + \delta UBB^T\delta U^T + \delta UB\delta B^T U^T + U\delta BB^T\delta U^T\Big).$$

We may further assume that the coding errors δU and δB are zero-mean independent random variables. Using the fact that $\mathbf{tr}(AB) = \mathbf{tr}(BA)$, the expected squared error becomes

$$\mathbb{E}\big(\mathbf{tr}\big(\delta W\delta W^T\big)\big) = \mathbb{E}\big(\mathbf{tr}\big(\delta B\delta B^T\big)\big) + \mathbb{E}\big(\mathbf{tr}\big(\Sigma^2\delta U^T\delta U\big)\big).$$

Now, let us encode each entry b_{ij} with a precision

$$\varepsilon' = \frac{\varepsilon}{\sqrt{n}}$$

and u_{ij} with a precision

$$\varepsilon_j'' = \frac{\varepsilon\sqrt{m}}{\sqrt{\lambda_j}n},$$

where λ_j is the jth eigenvalue of WW^T.[22] This is equivalent to assuming that the error δb_{ij} is uniformly distributed in the interval

$$\left[-\frac{\varepsilon}{\sqrt{n}}, \frac{\varepsilon}{\sqrt{n}}\right]$$

and δu_{ij} is uniformly distributed in the interval

$$\left[-\frac{\varepsilon\sqrt{m}}{\sqrt{\lambda_j}n}, \frac{\varepsilon\sqrt{m}}{\sqrt{\lambda_j}n}\right].$$

Under such a coding precision, it is easy to verify that

$$\mathbb{E}\big(\mathbf{tr}\big(\delta W\delta W^T\big)\big) \le \frac{2\varepsilon^2 m}{3} < \varepsilon^2 m. \tag{20}$$

Then, the mean squared error per vector in W is

$$\frac{1}{m}\mathbb{E}\big(\mathbf{tr}\big(\delta W\delta W^T\big)\big) < \varepsilon^2. \tag{21}$$

The number of bits needed to store the coordinates b_{ij}, with precision $\varepsilon' = \frac{\varepsilon}{\sqrt{n}}$, is

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{1}{2}\log_2\left(1 + \left(\frac{b_{ij}}{\varepsilon'}\right)^2\right) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\log_2\left(1 + \frac{b_{ij}^2 n}{\varepsilon^2}\right) \le \frac{m}{2}\sum_{i=1}^{n}\log_2\left(1 + \frac{n\sum_{j=1}^{m} b_{ij}^2}{m\varepsilon^2}\right) = \frac{m}{2}\sum_{i=1}^{n}\log_2\left(1 + \frac{n\lambda_i}{m\varepsilon^2}\right).$$

In the above inequality, we have applied the following inequality:

22. Notice that ε_j'' normally does not increase with the number of vectors m, because λ_j increases proportionally to m.

$$\frac{\log(1 + a_1) + \log(1 + a_2) + \cdots + \log(1 + a_n)}{n} \le \log\Big(1 + \frac{a_1 + a_2 + \cdots + a_n}{n}\Big) \tag{22}$$

for nonnegative real numbers $a_1, a_2, \ldots, a_n \ge 0$.

Similarly, the number of bits needed to store the entries of the singular vectors u_{ij}, with precision $\varepsilon'' = \frac{\varepsilon\sqrt{m}}{\sqrt{\lambda_i}n}$, is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{2}\log_2\left(1 + \left(\frac{u_{ij}}{\varepsilon''}\right)^2\right) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\log_2\left(1 + \frac{u_{ij}^2 n^2\lambda_j}{m\varepsilon^2}\right) \le \frac{n}{2}\sum_{j=1}^{n}\log_2\left(1 + \frac{n^2\lambda_j\sum_{i=1}^{n}u_{ij}^2}{m\varepsilon^2}\right) = \frac{n}{2}\sum_{j=1}^{n}\log_2\left(1 + \frac{n\lambda_j}{m\varepsilon^2}\right).$$

Thus, for U and B together, we need a total of

$$L(W) = \frac{m+n}{2}\sum_{i=1}^{n}\log_2\left(1 + \frac{n\lambda_i}{m\varepsilon^2}\right) = \frac{m+n}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2}WW^T\Big). \tag{23}$$

We thus have proven the statement given in the beginning of this section: $L(W) = (m + n)R(W)$ gives a good upper bound on the number of bits needed to encode W.

Appendix B
Nonzero-Mean Distribution

In the above analysis, we have assumed that the given vectors $W = (w_1, w_2, \ldots, w_m)$ are zero mean. In general, these vectors may have a nonzero mean. In other words, the points represented by these vectors may lie in an affine subspace instead of a linear subspace.

In case W is not zero mean, let $\mu \doteq \frac{1}{m}\sum_{i=1}^{m} w_i \in \mathbb{R}^n$ and define the matrix

$$V \doteq \mu \cdot 1_{1\times m} = (\mu, \mu, \ldots, \mu) \in \mathbb{R}^{n\times m}. \tag{24}$$

Then, $\bar{W} \doteq W - V$ is a matrix whose column vectors have zero mean. We may apply the same coding scheme in Appendix A to $\bar{W}$.

Let $\bar{W} = U\Sigma V^T \doteq UB$ be the SVD of $\bar{W}$. Let δU, δB, and $\delta\mu$ be the error in coding U, B, and μ, respectively. Then, the error induced on the matrix W is

$$\delta W = \delta\mu \cdot 1_{1\times m} + U\delta B + \delta UB. \tag{25}$$

Assuming that δU, δB, and $\delta\mu$ are zero-mean independent random variables, the expected total squared error is

$$\mathbb{E}\big(\mathbf{tr}\big(\delta W\delta W^T\big)\big) = m\mathbb{E}\big(\delta\mu^T\delta\mu\big) + \mathbb{E}\big(\mathbf{tr}\big(\delta B\delta B^T\big)\big) + \mathbb{E}\big(\mathbf{tr}\big(\Sigma\delta U^T\delta U\big)\big). \tag{26}$$

We encode entries of B and U with the same precision as before. We encode each entry μ_i of the mean vector μ, with the precision $\varepsilon' = \frac{\varepsilon}{\sqrt{n}}$, and assume that the error $\delta\mu_i$ is a uniform distribution in the interval $[-\frac{\varepsilon}{\sqrt{n}}, \frac{\varepsilon}{\sqrt{n}}]$. Then, we have $m\mathbb{E}(\delta\mu^T\delta\mu) = \frac{m\varepsilon^2}{3}$. By using (20) for the zero-mean case, the total squared error satisfies

$$\mathbb{E}\big(\mathbf{tr}\big(\delta W\delta W^T\big)\big) \le \frac{m\varepsilon^2}{3} + \frac{2m\varepsilon^2}{3} = m\varepsilon^2. \tag{27}$$

Then, the mean squared error per vector in W is still bounded by ε^2

$$\frac{1}{m}\mathbb{E}\big(\mathbf{tr}\big(\delta W \delta W^T\big)\big) \leq \varepsilon^2. \tag{28}$$

Now, in addition to the $L(\bar{W})$ bits needed to encode U and B, the number of bits needed to encode the mean vector μ, with precision $\varepsilon' = \frac{\varepsilon}{\sqrt{n}}$, is

$$\begin{aligned}\sum_{i=1}^{n}\frac{1}{2}\log_2\left(1+\left(\frac{\mu_i}{\varepsilon'}\right)^2\right) &= \frac{1}{2}\sum_{i=1}^{n}\log_2\left(1+\frac{n\mu_i^2}{\varepsilon^2}\right)\\ &\leq \frac{n}{2}\log_2\left(1+\frac{\mu^T\mu}{\varepsilon^2}\right),\end{aligned} \tag{29}$$

where the last inequality is from (22).

Thus, the total number bits needed to store W is

$$L(W) = \frac{m+n}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2}\bar{W}\bar{W}^T\Big) + \frac{n}{2}\log_2\Big(1+\frac{\mu^T\mu}{\varepsilon^2}\Big). \tag{30}$$

Notice that, if W is actually zero mean, then we have $\mu = 0$, $\bar{W} = W$, and the above expression for $L(W)$ is exactly the same as before.

APPENDIX C
RELATION TO MULTIPLE-CHANNEL CAPACITY

In wireless communication, the relationship between m transmitters and n receivers is often modeled as a fading MIMO channel

$$y = Wx + z, \tag{31}$$

where $y, z \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$. z is a random vector that models the (additive) channel noise. It is often assumed that z has a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$. Then, the model is known as the additive white Gaussian noise (AWGN) channel.

It is known in multiple-channel communications [36] that, in the high signal-to-noise ratio (SNR) regime, the channel capacity is given as

$$C(W) \doteq \frac{1}{2}\log_2\det\left(I + \frac{P}{m\sigma^2}WW^T\right), \tag{32}$$

where P is the total transmission power of the m transmitters [36]. The ratio P/σ^2 is the common SNR at each receiving antenna.

We could not help but notice a striking resemblance between the coding rate $R(W)$ in (11) and the wireless channel capacity $C(W)$ in (32). Notice that the noise variance σ^2 corresponds to the (entrywise) mean squared error ε^2/n. The power P is often assumed to be a constant and we may normalize it to be 1. Then, the capacity becomes exactly the coding rate of W

$$C(W) = R(W) = \frac{1}{2}\log_2\det\Big(I + \frac{n}{m\varepsilon^2}WW^T\Big).$$

Thus, the concavity of the coding rate function $R^{s,\infty}(W,\Pi)$ (Theorem 3 in Section 4) suggests that an even higher channel capacity may be achieved by probabilistically assigning the transmitters into multiple groups. The capacity of such a probabilistic transmitting channel is a concave function in Π

$$C(W,\Pi) \doteq \sum_{j=1}^{k}\frac{\mathbf{tr}(\Pi_j)}{2m}\log_2\det\left(I + \frac{n}{\varepsilon^2\mathbf{tr}(\Pi_j)}W\Pi_j W^T\right)$$

which has a unique maximum (for any given k).

ACKNOWLEDGMENTS

This work was partially supported by the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Grants IIS-0347456 and DMS-0349019, NSF CRS-EHS-0509151, NSF CCF-TF-0514955, and the US Office of Naval Reasearch (ONR) Grant N00014-05-1-0633 The authors would like to thank Professor Bin Yu of the Statistics Department of the University of California, Berkeley, for pointing out relevant references about MDL and mixture distributions. They would also like to thank Dr. Allen Yang for his contribution to the application to image segmentation and the new segmentation results, shown in Fig. 12. Finally, they would like to thank the anonymous reviewers for their helpful suggestions, especially the first reviewer, for suggesting the penalty function for choosing ε described in Section 2.

REFERENCES

[1] A. Jain, M. Murty, and P. Flynn, "Data Clustering: A Review," *ACM Computing Surveys,* vol. 31, no. 3, pp. 264-323, 1999.
[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer, 2001.
[3] M. Tipping and C. Bishop, "Mixtures of Probabilistic Principal Component Analyzers," *Neural Computation,* vol. 11, no. 2, pp. 443-482, 1999.
[4] M.A.T. Figueiredo and A.K. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 3, pp. 381-396, Mar. 2002.
[5] R. Vidal, Y. Ma, and S. Sastry, "Generalized Principal Component Analysis (GPCA)," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 12, pp. 1-15, Dec. 2005.
[6] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory,* vol. 28, no. 2, pp. 129-137, Mar. 1982.
[7] E. Forgy, "Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classifications (Abstract)," *Biometrics,* vol. 21, pp. 768-769, 1965.
[8] R. Jancey, "Multidimensional Group Analysis," *Australian J. Botany,* vol. 14, pp. 127-130, 1966.
[9] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math., Statistics, and Probability,* pp. 281-297, 1967.
[10] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proc. IEEE,* vol. 86, no. 11, pp. 2210-2239, 1998.
[11] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell, "Distance Metric Learning, with Application to Clustering with Side Information," *Proc. Ann. Conf. Neural Information Processing Systems,* 2002.
[12] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering Appearances of Objects under Varying Illumination Conditions," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* 2003.
[13] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc.,* vol. 39, no. B, pp. 1-38, 1977.
[14] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions.* John Wiley & Sons, 1997.
[15] Z. Ghahramani and G.E. Hinton, "The EM Algorithm for Mixtures of Factor Analyzers," Technical Report CRG-TR-96-1, Dept. Computer Science, Univ. of Toronto, 1996.
[16] N. Ueda, R. Nakan, and Z. Ghahramani, "SMEM Algorithm for Mixture Models," *Neural Computation,* vol. 12, pp. 2109-2128, 2000.
[17] T. Cover and J. Thomas, "Elements of Information Theory," *Wiley Series in Telecomm.,* 1991.
[18] J. Rissanen, "Modeling by Shortest Data Description," *Automatica,* vol. 14, pp. 465-471, 1978.

[19] A. Barron, J. Rissanen, and B. Yu, "The Minimum Description Length Principle in Coding and Modeling," *IEEE Trans. Information Theory,* vol. 44, no. 6, pp. 2743-2760, 1998.

[20] M. Hansen and B. Yu, "Model Selection and the Principle of Minimum Description Length," *J. Am. Statistical Assoc.,* vol. 96, pp. 746-774, 2001.

[21] M. Madiman, M. Harrison, and I. Kontoyiannis, "Minimum Description Length versus Maximum Likelihood in Lossy Data Compression," *Proc. 2004 IEEE Int'l Symp. Information Theory,* 2004.

[22] K. Rose, "A Mapping Approach to Rate-Distortion Computation and Analysis," *IEEE Trans. Information Theory,* vol. 40, no. 6, pp. 1939-1952, 1994.

[23] H. Benson, "Concave Minimization: Theory, Applications and Algorithms," *Handbook of Global Optimization,* R. Horst and P.M. Pardalos, eds., 1994.

[24] J. Ward, "Hierarchical Grouping to Optimize and Objective Function," *J. Am. Statistical Assoc.,* vol. 58, pp. 236-244, 1963.

[25] S. Kamvar, D. Klein, and C. Manning, "Interpreting and Extending Classical Agglomerative Clustering Methods Using a Model-Based Approach," Technical Report 2002-11, Dept. Computer Science, Stanford Univ., 2002.

[26] J. Hamkins and K. Zeger, "Gaussian Source Coding with Spherical Codes," *IEEE Trans. Information Theory,* vol. 48, no. 11, pp. 2980-2989, 2002.

[27] D. Donoho, M. Vetterli, R. DeVore, and I. Daubechies, "Data Compression and Harmonic Analysis," *IEEE Trans. Information Theory,* vol. 44, no. 6, pp. 2435-2476, 1998.

[28] R.A. Horn and C.R. Johnson, *Matrix Analysis.* Cambridge Univ. Press, 1985.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge Univ. Press, 2004.

[30] Z. Ghahramani and G. Hinton, "The EM Algorithm for Mixtures of Factor Analyzers," Technical Report CRG-TR-96-1, Univ. of Toronto, 1996.

[31] Z. Ghahramani and M. Beal, "Variational Inference for Bayesian Mixtures of Factor Analyzers," *Advances in Neural Information Processing Systems,* vol. 12, pp. 449-455, 2000.

[32] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and Texture Analysis for Image Segmentation," *Int'l J. Computer Vision,* vol. 43, no. 1, pp. 7-27, 2001.

[33] S. Zhu, C. Guo, Y. Wu, and Y. Wang, "What Are Textons," *Proc. European Conf. Computer Vision,* pp. 793-807, 2002.

[34] G. Mori, "Guiding Model Search Using Segmentation," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 1417-1423, 2005.

[35] A. Yang, J. Wright, W. Hong, and Y. Ma, "Segmentation of Natural Images via Lossy Data Compression," Technical Report, Coordinated Science Laboratory, Univ. of Illinois, 2006.

[36] D. Tse and P. Viswanath, *Fundamentals of Wireless Comm.* Cambridge Univ. Press, 2005.

Yi Ma received bachelor degrees in automation and applied mathematics from Tsinghua University, Beijing, in 1995. He received the MS degree in electrical engineering and computer science (EECS) in 1997, the MA degree in mathematics in 2000, and the PhD degree in EECS in 2000, all from the University of California, Berkeley. Since 2000, he has been with the faculty of the Electrical and Computer Engineering Department, University of Illinois, Urbana-Champaign, where he now holds the rank of associate professor. His main research interests include systems theory and computer vision. He is a senior member of the IEEE, the IEEE Computer Society, and a member of the ACM. He was the recipient of the David Marr Best Paper Prize at the International Conference on Computer Vision in 1999 and Honorable Mention for the Longuet-Higgins Best Paper Award at the European Conference on Computer Vision in 2004. He received the CAREER Award from the US National Science Foundation in 2004 and the Young Investigator Program Award from the US Office of Naval Research in 2005.

Harm Derksen received the master's degree in mathematics from the University of Nijmegen, The Netherlands, in 1993 and the PhD degree in mathematics from the University of Basel, Switzerland, in 1997. After postdoctoral positions at Northeastern University and Massachusetts Institute of Technology (MIT), he joined the faculty of the mathematics department of the University of Michigan, Ann Arbor, in 2000, where he is now an associate professor. His main research areas are commutative algebra and invariant theory, but he has many other interests including computer vision and coding theory.

Wei Hong received the BE degree in information science and technology from Xi'an Jiaotong University, Xi'an, China, in 1999. He received the MS degree in electrical engineering in 2003 and the PhD degree in electrical and computer engineering in 2006, both from the University of Illinois, Urbana-Champaign. He is currently a member of technical staff of DSP Solutions Research and Development Center, Texas Instruments, Dallas. His research interests include image and video analysis, computer vision, and machine learning. He is a member of the IEEE.

John Wright received the bachelor's degree in computer engineering in 2004 and the MS degree in electrical engineering in 2006, both from the University of Illinois, Urbana-Champaign. He is currently a PhD candidate in the Department of Electrical and Computer Engineering, University of Illinois, and a research assistant at the Coordinated Science Laboratory. His research interests include data segmentation, image representation, and computer vision. He is a student member of the IEEE.