Analyzing hyperspectral images into multiple subspaces using Gaussian mixture models

Clay D. Spence

SRI International, 201 Washington Road, Princeton, NJ, USA

ABSTRACT

I argue that the spectra in a hyperspectral datacube will usually lie in several low-dimensional subspaces, and that these subspaces are more easily estimated from the data than the endmembers. I present an algorithm for finding the subspaces. The algorithm fits the data with a Gaussian mixture model, in which the means and covariance matrices are parameterized in terms of the subspaces. The locations of materials can be inferred from the fit of library spectra to the subspaces. The algorithm can be modified to perform material detection. This has better performance than standard algorithms such as ACE, and runs in real time.

Keywords: Hyperspectral, HSI, unmixing, subpace, Gaussian mixtures

1. INTRODUCTION

Many hyperspectral processing algorithms assume the linear mixing model $\mathbf{X} = \mathbf{AS} + \mathbf{N}$, with the following definitions.

- $\mathbf{X}$ is the data, for which the i, λ-th element is some measure of intensity at the i-th pixel in the λ-th spectral band. The quantity represented is typically the radiance or reflectance.
- $\mathbf{A}$ is the abundance matrix, for which the i, m-th element is the fraction of pixel i's spectrum accounted for by material m's spectrum.
- $\mathbf{S}$ is the matrix of material spectra, for which the m,λ-th element is the relative amplitude of the m-th material in the λ-th band.
- $\mathbf{N}$ is the matrix of noise in the observations, for which the i, λ-th element is the noise in the i-th pixel's λ-th band.

The rows of $\mathbf{S}$ are the *endmembers*, the spectra of pure samples of the materials in the scene.

If the endmembers are assumed known, one typically performs a least-squares fit to estimate $\mathbf{A}$ from $\mathbf{X}$ and $\mathbf{S}$, thereby giving abundances of materials at each pixel. This can be used for a variety of tasks, such as material detection. However, it is often the case that a library of spectra does not adequately fit a new scene, due to the presence of materials not in the library, and a variety of effects that cause a material's observed spectrum to differ from the library spectrum. These effects can include chemical or physical differences between the laboratory samples and the material in the field, non-linear effects including interactions between endmembers, and probably others. It is often assumed that these effects can be handled by introducing extra endmembers, so that the linear mixing model remains useful. If we do not assume that we know the endmembers, we can attempt to estimate them from the data. This has been called the *blind unmixing* problem.

E-mail: clay.spence@sri.com, Telephone: 1 609 734 3039, Fax: 1 609 734 2662, sri.com

Original available at http://dx.doi.org/10.1117/12.2224234.

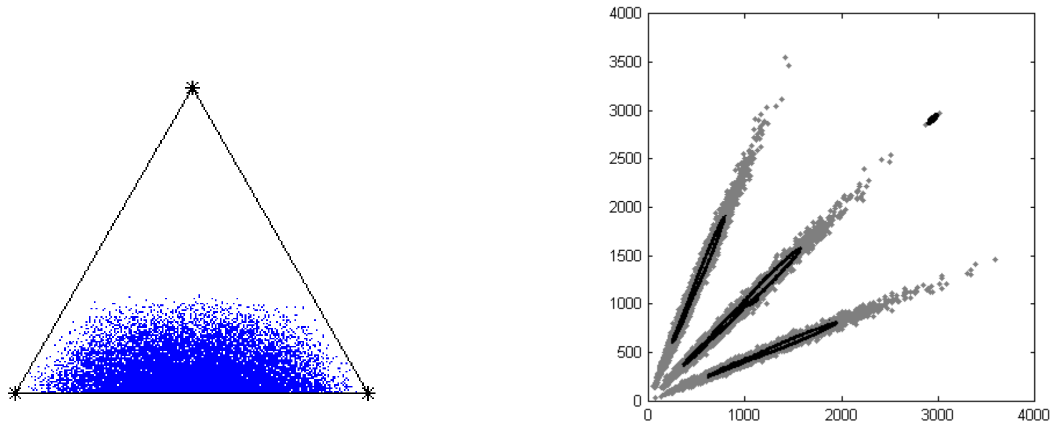

Figure 1. Potential problems with most unmixing models. (Left) Endmembers that are poorly determind by the data. The points indicate where the vectors pierce a slice through the cone formed by the endmembers in spectral space. The vertices of the triangle are the true endmembers. If an endmember is only present in small quantities, it may not be well-determined by the data. (See Section 3.1.1.) (Right) Most unmixing algorithms assume fewer endmembers than spectral bands, a condition that is easily violated. Yet it is possible to fit useful subspaces to the data. The superimposed ellipses are results of the MSG algorithm (Section 3.1.2).

A variety of blind unmixing algorithms have been proposed, including ICA of various sorts,[1–3] more purely geometric principles,[4] and non-negative matrix factorization.[5] Most of these use the fact that the data points are *convex* mixtures of the endmembers, that is, the coefficients are non-negative, so all data points must lie within a polytopic cone in the space of spectra, with vertex at the origin and edges that are the endmembers. Since noise perturbs the data vectors, they lie close to the cone rather than strictly inside it. The algorithms then attempt to find a cone that tightly fits the data.

For many data sets blind unmixing will be ill-conditioned. To see this, suppose there are no data vectors that are nearly-pure examples of a particular endmember. The endmember might still be well-determined if the cone faces that intersect at the endmember are well-determined by the data, but it isn't hard to imagine situations in which this is not the case. For example, suppose a cone has three endmembers, but two of them dominate, while the third always occurs in small amounts, never more than the least of the other two (Figure 1, left). Worse situations are imaginable, in which no pixels lie near the cone's boundary. It is even possible for data to support endmembers that are false, if certain proportions are common.[*] It will be more common that some endmembers are well-defined by the data while others are not. Unmixing algorithms need to account for the dimensions of the data created by those endmembers, but their estimates will be poor. Thus we should not be surprised if the blind unmixing problem is not completely solvable in many real cases.

Another issue with blind unmixing is the constraint that the number of endmembers be less than the number of spectral bands. This arises because, in order to be well-determined, $\mathbf{A}$ and $\mathbf{S}$ must have fewer parameters than the number of observed values in $\mathbf{X}$. If there are N pixels and L spectral bands, and we assume that M endmembers are present, we need $M < NL/(N + L)$. Since N is usually much greater than L, this condition is approximately $M < L$. This will be violated in many situations unless the image covers a small area. Situations in which it may be violated include urban and suburban scenes, since they can contain a wide variety of natural and artificial materials.

If not the endmembers, what can we extract from the data? I propose that we can robustly estimate a set of low-dimensional subspaces,[†] each spanned by a subset of endmembers. To motivate this, suppose that we could

[*]In such a case we could instead question what we should define as the endmembers.

[†]Some sources use the term "subspace" to refer to a subset of the coordinates of the full space. This is *not* the sense in which I am using the term. I am referring to the standard mathematical notion of a vector space that is a subset of a larger vector space.

segment a hyperspectral image, as suggested by Kettig and Landgrebe.[6] They argued that each segment would contain common materials. In each segment we can safely assume that there are relatively few such materials. For example, patches of grass, road, and roof are each likely to have few endmembers. Even if some or all of a segment's endmembers are difficult to estimate from its pixels, those pixels span the same subspace that is spanned by the endmembers. We should be able to find the subspace corresponding to the segment's pixels using fairly standard PCA-like methods. Only dimensions spanned by endmembers with very low abundances, obscured by noise, will be difficult to estimate. This multiple-subspace estimation problem thus avoids issues with endmembers that are poorly-determined by the data. In addition, it is applicable to scenes with more endmembers than dimensions, as long as each segment has relatively few endmembers (Figure 1, right).

Estimates of the image's subspaces may not be all we would like, but they are useful. For example, we can decide whether a material is present in a segment by computing the angle between the material's library spectrum and the segment's subspace. If the angle is small enough we can conclude that the material is present. Another example is segmentation of the scene by subspace membership. This groups pixels in an unsupervised manner into general classes. Semantic labels can then be assigned to the discovered classes.

Below I present an algorithm for the multiple subspace estimation problem. It fits the data with a Gaussian mixture model, in which each mixture component's covariance matrix is constrained to have a large variance within a subspace and a small variance perpendicular to it.[‡] Each component has its own subspace. Accordingly we will refer to this model as the Mixture of Subspace Gaussians, or *MSG*, model. The E-step segments the scene, while the variances and subspaces are fit in the M-step, including the number of dimensions of each subspace. I present two variants. The first is simpler, since it assumes zero mean, while the second better fits the data by having non-zero means that are constrained to lie within their components' subspaces. Unlike Kettig and Landgrebe,[6] the proposed method does not use spatial information. Although that should improve performance it is not the focus here. In addition I present a modified MSG algorithm for explicit material detection.

2. MSG MODELS

For both variants, use $\mathbf{P}$ to denote an orthogonal projection matrix onto a subspace, and $\overline{\mathbf{P}}$ for the projection matrix onto the complement subspace, so that $\overline{\mathbf{P}} = \mathbf{I} - \mathbf{P}$. Recall that an orthogonal projection matrix $\mathbf{P}$ must be symmetric and idempotent, that is, $\mathbf{P}^T = \mathbf{P}$ and $\mathbf{P}^2 = \mathbf{P}$. Every vector $\mathbf{v}$ can be split into a piece $\mathbf{v}_\parallel$ within the subspace onto which $\mathbf{P}$ projects, and a piece $\mathbf{v}_\perp$ orthogonal to that subspace, so $\mathbf{v}_\parallel = \mathbf{P}\mathbf{v}$, and $\mathbf{v}_\perp = \mathbf{v} - \mathbf{v}_\parallel = (\mathbf{I} - \mathbf{P})\mathbf{v} = \overline{\mathbf{P}}\mathbf{v}$. It is easy to see that $\mathbf{v}_\parallel^T \mathbf{v}_\perp = 0$. For brevity we'll often refer to such matrices simply as "projection matrices," since we do not consider oblique projections, for which the matrix is not symmetric.

2.1 Models with zero means

For data points $\mathbf{x}$ in L dimensions write the probability model as

$$\Pr(\mathbf{x}) = \sum_c \Pr(c) \Pr(\mathbf{x} \,|\, c), \qquad \text{with} \tag{1}$$

$$\Pr(\mathbf{x} \,|\, c) = (2\pi)^{-L/2} |\boldsymbol{\Sigma}_c|^{-1/2} \exp\!\left(-\tfrac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}_c^{-1} \mathbf{x}\right), \qquad \text{and} \tag{2}$$

$$\boldsymbol{\Sigma}_c = \sigma_{c\parallel}^2 \mathbf{P}_c + \sigma_{c\perp}^2 \overline{\mathbf{P}}_c, \tag{3}$$

Here $\sigma_{c\parallel}^2$ is the model variance within the subspace, and $\sigma_{c\perp}^2$ is the model variance perpendicular to it. $\mathbf{P}_c$ projects onto the component's principal subspace. This subspace, including its dimension, are determined by the EM algorithm[8] for this model, as follows.

The E-step and update of the priors (mixture weights) $\Pr(c)$ are the same as for any GMM, but the update of the covariance matrices is not. The derivation is given in Appendix A. Write the posteriors as $\Pr(c \,|\, \mathbf{x}_i, \Theta')$,

[‡]See Beaven et al.[7] for a rather different application of Gaussian mixture models and subspace decomposition to hyperspectral data. Note that they use a single principal subspace.

where Θ' is the value of the model parameters from the previous M-step, or the initialization. Define

$$Z_c = \sum_i \Pr(c \,|\, \mathbf{x}_i, \Theta') \qquad \text{and} \tag{4}$$

$$\mathbf{R}_c = Z_c^{-1} \sum_i \Pr(c \,|\, \mathbf{x}_i, \Theta') \mathbf{x}_i \mathbf{x}_i^T, \tag{5}$$

the latter being the correlation matrix for the c-th component. The M-step is then

1. $\Pr(c) \leftarrow Z_c \big/ \sum_{c'} Z_{c'}$.
2. Compute $\mathbf{R}_c$ and its eigenvalues $\lambda_{c,i}$ and eigenvectors $\mathbf{v}_{c,i}$, with eigenvalues in descending order.
3. For each c, pick the values for the c-th component's subspace dimension M_c that maximizes

$$\tilde{q}_c = -M_c \log \sigma^2_{c\|} - (L - M_c) \log \sigma^2_{c\perp}, \qquad \text{where} \tag{6}$$

$$\sigma^2_{c\|} = \frac{1}{M_c} \sum_{i=1}^{M_c} \lambda_{c,i} \qquad \text{and} \tag{7}$$

$$\sigma^2_{c\perp} = \frac{1}{L - M_c} \sum_{i=M_c+1}^{L} \lambda_{c,i}. \tag{8}$$

   Keep these variances corresponding to the optimal M_c's.

4. For each c, set the c-th component's basis matrix for its subspace to the principal M_c eigenvectors of $\mathbf{R}_c$

$$\mathbf{V}_c = \left(\mathbf{v}_{c,1} \ldots \mathbf{v}_{c,M_c}\right), \qquad \text{and let} \tag{9}$$

$$\mathbf{P}_c = \mathbf{V}_c \mathbf{V}_c^T. \tag{10}$$

The quantity $\tilde{Q} = \sum_c Z_c q_c$ is the expected log of the complete-data likelihood, after inserting (7)–(10), and leaving out those terms that do not depend on the choice of the M_c's. Since $\tilde{q}_c$ depends only on M_c and the eigenvalues of $\mathbf{R}_c$, the search for the optimal M_c's can be quite fast. In the experiments below greedy search was used, starting with $M_c = 1$ for all c, and iterating. In each iteration the algorithm increased and decreased each M_c by one to check for larger $\tilde{q}_c$.

2.2 Non-zero means

The derivation is in Appendix B. To accommodate non-zero means, (2) changes to

$$\Pr(\mathbf{x} \,|\, c) = (2\pi)^{-L/2} |\boldsymbol{\Sigma}_c|^{-1/2} \exp\left[-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right] \tag{11}$$

where $\boldsymbol{\mu}_c$ is the model mean for component c. This is constrained to lie in the component's subspace, so that $\overline{\mathbf{P}}_c \boldsymbol{\mu}_c = \mathbf{0}$. Again, it is only the M-step that is unusual. Define Z_c and $\mathbf{R}_c$ as before, and additionally denote the data means and covariance matrices by

$$\mathbf{m}_c = Z_c^{-1} \sum_i \Pr(c \,|\, \mathbf{x}_i, \Theta') \mathbf{x}_i, \qquad \text{and} \tag{12}$$

$$\mathbf{C}_c = Z_c^{-1} \sum_i \Pr(c \,|\, \mathbf{x}_i, \Theta') (\mathbf{x}_i - \mathbf{m}_c)(\mathbf{x}_i - \mathbf{m}_c)^T \tag{13}$$

$$= \mathbf{R}_c - \mathbf{m}_c \mathbf{m}_c^T. \tag{14}$$

For each component c, and a given number of dimensions M_c for its subspace, begin by choosing some initial basis $\mathbf{V}_c$ and variances. Then iterate, alternating between computing the variances given the basis and the basis given the variances. The variances are given by

$$\sigma^2_{c\|} = \operatorname{Tr}(\mathbf{P}_c \mathbf{C}_c) \big/ \operatorname{Tr} \mathbf{P}_c \qquad \text{and} \tag{15}$$

$$\sigma^2_{c\perp} = \operatorname{Tr}(\overline{\mathbf{P}}_c \mathbf{R}_c) \big/ \operatorname{Tr} \overline{\mathbf{P}}_c. \tag{16}$$

With these values, compute $\mathbf{V}_c$ as the M_c principal eigenvectors of $(\sigma_{c\perp}^{-2} - \sigma_{c\|}^{-2})\mathbf{R}_c + \sigma_{c\|}^{-2}\mathbf{m}_c\mathbf{m}_c^T$. Iterate until the change in the variances is sufficiently small. Note that this iteration is done within an M-step.

To find the optimal values of M_c, again search for the value that maximizes $\tilde{q}_c$ at the optimal $\mathbf{V}_c$ and variances for each M_c. As for the zero-mean model, in the experiments I start with $M_c = 1$ and perform a greedy search. With the optimal $\mathbf{V}_c$ and variances for a given M_c, $\tilde{q}_c$ has the same form, given by (6). Finally, project the data means onto their components' subspaces to get the model means, i.e.,

$$\boldsymbol{\mu}_c = \mathbf{P}_c\mathbf{m}_c. \tag{17}$$

2.3 Model complexity

To choose the number of MSG components I used a modification of the model-growing algorithm of Verbeek, et al.[9] This semi-randomly creates several new trial components from current components, quickly estimates the potential improvement each would give, keeps the best, and uses EM to refine all parameters of the new model. I used the minimum description length (MDL) criterion to decide when to stop growing the model.

I modified the algorithm of Verbeek et al. for creating trial components, in order to encourage the algorithm to find new subspaces. Their algorithm takes the subset $\mathcal{S}_c$ of data most likely to belong to the component being split (c), and selects two data points, say $\mathbf{x}_1$ and $\mathbf{x}_2$, at random from $\mathcal{S}_c$. It then splits $\mathcal{S}_c$ into two subsets, those closer to either $\mathbf{x}_1$ or $\mathbf{x}_2$. Two trial components are initialized by fitting each to one of these subsets. For half of the trial components I kept this procedure, while for the other half I formed a trial component by randomly selecting one element $\mathbf{x} \in \mathcal{S}_c$. The trial component was initialized by setting its mean to $\mathbf{x}$, it's subspace to a single dimension with basis $\mathbf{x}/\|\mathbf{x}\|$, and variances the same as the parent component c.

The MDL cost is $-\log \mathcal{L} + (d/2)\log N$, where $\mathcal{L}$ is the likelihood of the training data under the model, N is the number of training data examples, and d is the number of parameters in the model. To get d we add the number of parameters in $\Pr(c)$ and the sum over components of the number of parameters in each. If the number of components is C, there are $C - 1$ parameters for the $\Pr(c)$, since $\sum_c \Pr(c) = 1$. The number of parameters for a projection matrix $\mathbf{P}_c$ is the number needed to specify a basis, $\mathbf{V}_c$. This has LM_c elements, but it is orthonormal, so the symmetric expression $\mathbf{V}_c^T\mathbf{V}_c = \mathbf{I}_{M_c}$ represents $M_c(M_c + 1)/2$ constraints. In addition, we can apply an arbitrary rotation $\mathbf{Q}$ to $\mathbf{V}_c$, so in coding $\mathbf{V}_c$ we can reduce the number of parameters by the number in $\mathbf{Q}$, which is $M_c(M_c - 1)/2$ since it is orthogonal. All of this leaves $M_c(L - Mc)$ parameters for $\mathbf{V}_c$. The mean is constrained to lie in the subspace, so it has M_c parameters. Finally there are the two variance parameters for each component. Summed together we get

$$d = 3C - 1 + \sum_{c=1}^{C}\big[M_c(L - M_c + 1)\big]. \tag{18}$$

2.4 Computational issues

An implementation never has to compute the projection matrices, since the E-step only needs $\mathbf{x}^T\mathbf{P}_c\mathbf{x}$ and $\mathbf{x}^T\overline{\mathbf{P}}_c\mathbf{x}$, which can be computed as $\|\mathbf{V}_c\mathbf{x}\|^2$ and $\|\mathbf{x}\|^2 - \|\mathbf{V}_c\mathbf{x}\|^2$. Therefore we only need to store $\mathbf{V}_c$, and only need to compute the squared-magnitude of $\mathbf{V}_c\mathbf{x}$, which has relatively few dimensions. In addition, the $\|\mathbf{x}\|^2$ can be pre-computed once before all iterations. Other projections onto a subspace or its complement can also be performed with $\mathbf{V}_c$. They take up much less time, since there are many fewer of them than the number of data vectors.

The computational expense of MSG model fitting is usually dominated by the cost of computing the correlation matrices, $\mathbf{R}_c$, in the M-step. With N data vectors, L spectral bands, and C mixture components, it takes order CNL^2 operations to compute these. By contrast the eigen-decompositions take order CL^3 operations, so the ratio is N/L, which is usually large. There is also a tendency for the time to increase faster than linearly with C, since more iterations are required as C increases.

2.5 Target detection

Many algorithms for detecting targets in spectral data use a single spectrum to characterize the target material, but some[10–12] use a subspace with multiple dimensions. This can be an advantage, since it allows for variations in the spectrum of the material of interest. Whether or not they use a multi-dimensional subspace, most use a single Gaussian to model the background. While this makes analysis of the algorithm tractable, performance could be improved by taking the non-Gaussian structure of the data into account. MSG models enable this by modeling the background with multiple subspaces. A simple approach is to fix the subspace of one MSG component to that of the target, and let the other components adaptively model the background. The posterior probability of a pixel belonging to the target component is the obvious score to use for detection. It is easy to extend this to multiple target materials by including a fixed subspace for each.

If we fix the MSG model to have one component for the background and one for the target material, it resembles algorithms in the literature. To see this, re-write the posterior as

$$\Pr(c_t \,|\, \mathbf{x}) = \frac{\Pr(\mathbf{x} \,|\, c_t) \Pr(c_t)}{\Pr(\mathbf{x} \,|\, c_t) \Pr(c_t) + \Pr(\mathbf{x} \,|\, c_b)(1 - \Pr(c_t))} \tag{19}$$

$$= \left[1 + \frac{(1 - \Pr(c_t)) \Pr(\mathbf{x} \,|\, c_b)}{\Pr(c_t) \Pr(\mathbf{x} \,|\, c_t)}\right]^{-1}, \tag{20}$$

where c_t and c_b are the target and background class labels, respectively. The second term in the brackets is the inverse of the odds of belonging to the target class. Inserting the form for the Gaussian class distributions lets us compute the log-odds as

$$\text{logOdds}(c_t \,|\, \mathbf{x}) = \text{const} - \frac{\sigma_{c\perp}^{-2} - \sigma_{c\|}^{-2}}{2} \mathbf{x}^T (\mathbf{P}_{c_b} - \mathbf{P}_{c_t}) \mathbf{x}. \tag{21}$$

The final term is similar to some of the detection functions in Manolakis, et al.,[10] up to multiplicative and additive constants. (Note that their $\mathbf{P}_A^\perp$ is $\overline{\mathbf{P}}_A$ here, and $\mathbf{P}_{c_b} - \mathbf{P}_{c_t} = \overline{\mathbf{P}}_{c_t} - \overline{\mathbf{P}}_{c_b}$.)

Besides the multiple subspaces, the MSG approach differs from the subspace algorithms described in Manolakis, et al.[10] in the source data for the background estimate. Both use some portion of the scene for this. Manolakis et al. emphasize the need to avoid contaminating the background estimate with target pixels. This is often done by assuming that a small test region contains the target material, then this is surrounded by a larger "guard" region, whose pixels are not used. The background Gaussian is estimated from pixels outside of the guard and test regions. This is intended for small targets, but for some problems the material can cover a relatively large region, such as in inspection of nearby surfaces. The MSG algorithms separates the background through the E-step, which segments the scene for us, and so can deal with large patches of target material in the scene.

To achieve adequate speed using MSGs for material detection, I suggest a pair of techniques. First, fix the number of background subspaces to two, so with one material to be detected there are three subspaces in the model. This goes beyond the Gaussian background assumption, though the form for each of the two background components is simplified compared to the usual single Gaussian background model. Second, process one line at a time, and keep the results from previous lines to analyze subsequent lines. In particular the background subspaces usually do not need to be re-estimated. Occasionally the resulting fit will be poor, indicating a new background material in the new line, so the background subspaces should be re-estimated. With these approaches I have implemented versions that run in real time on computers with modest computational power.

3. EXPERIMENTS

To demonstrate the MSG algorithm's features, I first describe experiments with artificial data for the unsupervised subspace fitting problem, then the same algorithm with real data. I then describe an experiment with target detection in artificial data.

3.1 Unsupervised analysis

3.1.1 Artificial data with a poorly-defined endmember

To demonstrate the MSG algorithm's ability to handle endmembers with at most low abundance in any pixel, I generated a 128-by-128 pixel image with random mixtures of three spectra from USGS' speclib04y library.[13] These have 224 bands. I used "Alunite GDS82 Na82", "Montmorillonite+Illi CM37", and "Desert_Varnish GDS141". I used the desert varnish spectrum as the low-abundance endmember.

At each pixel the abundance r_{al} of alunite *relative to* montmorillonite was drawn from a beta distribution $\Pr(x) = x^{\alpha-1}(1-x)^{\beta-1}/\mathrm{B}(\alpha,\beta)$, where $\mathrm{B}(\alpha,\beta)$ is the beta function, and I chose $\alpha = \beta = 3$ to give a density that decreases smoothly to zero at the endpoints, zero and one. The abundance of montmorillonite relative to alunite was $r_{\mathrm{mont}} = 1 - r_{\mathrm{al}}$. The *absolute* abundance of desert varnish at the pixel was chosen to be one-third of a value drawn from a beta distribution with $\alpha = 1$ and $\beta = 3$. This gives abundances with values more likely near 0, decreasing smoothly to zero probability at 1/3. The absolute abundances of alunite and montmorillonite were then computed as their relative abundances multiplied by one minus the abundance of desert varnish. The resulting absolute abundances sum to one. They are illustrated in the left plot of Figure 1. To give the spectra a range of magnitudes, I scaled the three mineral spectra to have unit norm, then mixed using these abundances, and the resulting spectrum was multiplied by a value drawn from a gamma distribution $\Pr(x) = x^{k-1}e^{-x/\theta}/(\Gamma(k)\theta^k)$. I chose $k = 4$, which somewhat suppresses values near zero, and $\theta = 10^6$. Since I had normalized the mineral spectra by dividing by their norms, the typical resulting clean pixel spectrum has a maximum over the bands of between 10^4 and 5×10^5. Noise was introduced by using the values as the means of Poisson distributions, from which the final pixel values were drawn.

The resulting MSG model with non-zero means had six components. The means of the components were all within 0.054° of each other, and differed in magnitude (norm) from 1.3×10^6 to 7.5×10^6. All of the subspaces were three-dimensional. The largest angle between any of the MSG's estimated subspaces is 0.74°. Apparently the MSG model used multiple components to fit the different magnitudes in the data, but they all have essentially the same subspace.

For comparison, I ran the same data through the unmixing functions in the Matlab® hyperspectral toolbox,[14] which implement the ATGP,[15] ICA-endmember extraction (ICA-EEA), VCA,[4] and PPI algorithms, and also PPI preceded by MNF.[16] Note that all of these are explicitly designed for data in which all endmembers appear in pure form. All were given the true number of endmembers (three) as input. The angles between the extracted endmembers and the three true endmembers are given in Table 1. For MSG, the angles given are the maxima over the estimated subspaces of their angles with the true endmember of the given material. For the other algorithms, the angle given is the minimum over the endmembers estimated by the algorithm and the given true endmember. For comparison, the angle between the alunite and montmorillonite spectra is 12.0°, that between the alunite and desert varnish spectra is 13.1°, and that between the montmorillonite and desert varnish spectra is 23.8°. We can conclude that the MSG model finds good subspaces in this case, while the other algorithms fail to estimate the desert varnish spectrum.

This comparison may be thought unfair, as MSG was not estimating endmembers. Still the example demonstrates that in some cases finding endmembers can be difficult while the MSG algorithm can find the relevant subspaces with good accuracy. But this difference in the goal of the algorithms needs to be kept in mind while comparing numbers.

3.1.2 Artificial data with more endmembers than bands

To demonstrate the MSG algorithm's ability to handle data with more endmembers than bands, I constructed a datacube with only two bands, to make it trivial to visualize, and three endmembers. The endmembers were unit vectors that made angles of 22.5°, 45°, and 67.5° with the x-axis. I drew random abundances for each endmember at each pixel from a Gamma distribution with parameters $k = 6$ and $\theta = 200$. These were used as means of Poisson distributions from which the final pixel values were drawn.

I compared the MSG results with those for the Matlab HyperspectralToolbox[14] algorithms, described above. Note that the ICA-EEA function only returned two endmembers, and the VCA function gave an error if asked to return three, so I requested two. The resulting errors in the estimates of the endmember angles are shown

Table 1. Errors of algorithms when one endmember is never pure. For MSG, this is the maximum (see text) of the angles between the subspaces and the true endmember. For the other algorithms, this is the angle between the closest of the estimated endmembers and the true endmember for the material. This is for artificial data with three endmembers: alunite, montmorillonite, and desert varnish. Desert Varnish never appears with an abundance above one-third. All angles are in degrees.

Material\Algorithm	AGTP	ICA-EEA	VCA	PPI	MNF/PPI	**MSG**
Alunite	1.79	8.01	0.58	4.71	4.71	**0.017**
Montmorillonite	2.70	3.70	0.12	4.56	1.53	**0.018**
Desert Varnish	13.34	16.01	12.21	16.87	12.79	**0.071**

Table 2. Errors of algorithms for data with more endmembers than spectral bands. This is for artificial data with two bands and endmembers at 22.5°, 45°, and 67.5° to the x-axis. Algorithms with a dash in one entry computed two instead of three endmembers, so the largest error is not shown. All angles are in degrees.

True angle\Algorithm	AGTP	ICA-EEA	VCA	PPI	MNF/PPI	**MSG**
22.5	**0.40**	3.72	—	9.34	**0.40**	1.53
45	**0.41**	—	0.28	0.57	**0.41**	1.48
67.5	0.73	4.40	16.50	9.92	0.73	**0.02**

in Table 2. The numbers for the MSG algorithm are obtained by finding all of the components that are closer to the correct angle than the other two angles, then taking the maximum of the absolute differences over these components. The AGTP and MNF/PPI algorithms behave well with more endmembers than bands, giving more accurate angles than MSG for two of the three subspaces. The MSG algorithm is at a slight disadvantage here, since the given error is for one of the components that were close to the correct subspace, and it is the largest error of those components. The AGTP and MNF/PPI algorithms could use all of the points, or those most distant from the origin.

3.1.3 AVIRIS Cuprite data

I applied the MSG models to a part of the AVIRIS Cuprite collection from 1997 (Figure 2).[§] I selected a 190-by-250 pixel region to approximately match that used in Nascimento and Dias.[4] To avoid bands dominated by noise and those spatially shifted with respected to the majority of bands, I used bands 6–105, 116–150, and 170–185. The processed patch and resulting most likely components for each pixel are shown in Figure 2. The MSG algorithm found nineteen components, hence subspaces. These had dimensions ranging from seven to sixteen.

It is somewhat difficult to compare this with the unmixing algorithms, since their goals are different. One approach is to compare what might be called the reconstruction error. I computed this for two unmixing algorithms, VCA and non-negative matrix factorization (NNMF).[5,17] In principle, the latter does not need pure pixels for all of the endmembers. For the unmixing algorithms, I computed $\|\mathbf{x}-\mathbf{aS}\|$, using the L^2-norm, averaged over pixels. For MSG I computed the distance from each pixel to its best fit subspace, and averaged over pixels. The results are shown in Table 3. Error maps, i.e., the reconstruction errors at each pixel, are shown in Figure 3. The MSG algorithm gives a significantly better fit to the data.

Table 3. Reconstructions errors of algorithms on the Cuprite data set.

Algorithm	Reconstruction error
VCA	720.9
NNMF	659.8
MSG	**137.6**

[§]Downloaded from `ftp://popo.jpl.nasa.gov/pub/free_data/f970619t01p02r02c_rfl.tar`.

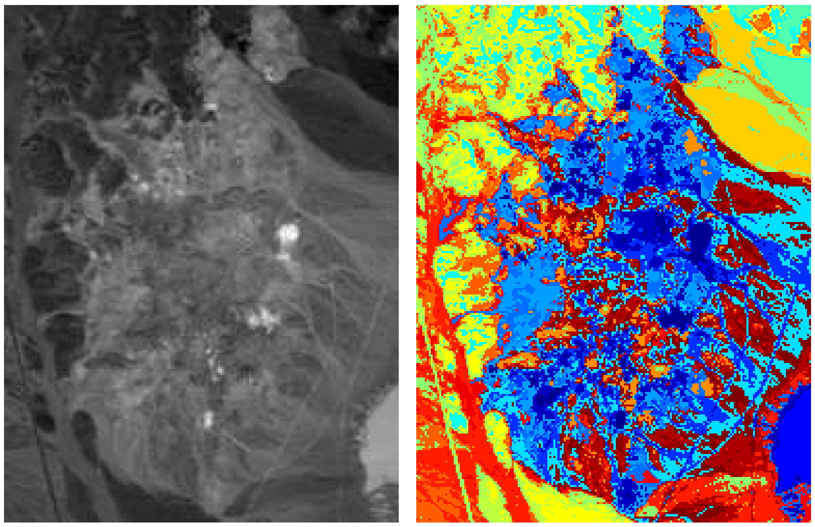

Figure 2. *Left*: Band 100 of portion of Cuprite 1997 AVIRIS reflectance data used in experiment. *Right*: Most likely MSG model components for each pixel. Each color is one of the 19 components.

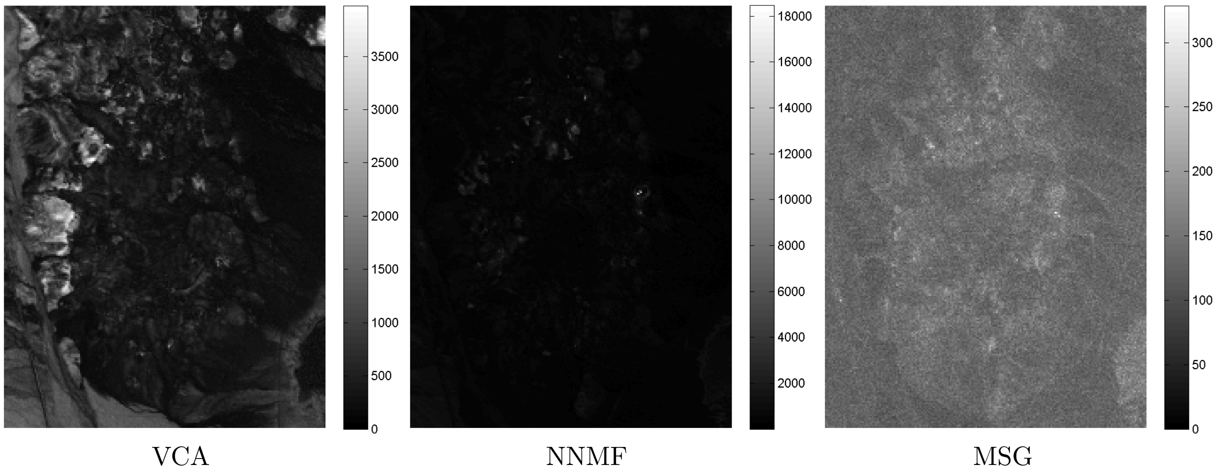

Figure 3. Reconstruction errors of three algorithms on AVIRIS Cuprite 1997 data. Note the difference in the scales. For VCA and NNMF this is the L^2 error between the original pixel and $\mathbf{aS}$, where $\mathbf{a}$ and $\mathbf{S}$ are the estimated abundance and endmember set, respectively. For MSG this is the distance to the subspace of the pixel's most-likely component.

3.2 Target detection

To demonstrate the modified MSG model on a target detection problem, I generated a 128-by-128 pixel artificial image. Pixels in the left half of the image were mixtures of the six Alunite spectra in the USGS speclib04y library,[13] while those in the right half are mixtures of the ten Montmorillonite spectra. Each of these sixteen endmembers were first scaled to have unit norm. For the target spectrum I chose the sum of the mean of the Alunite spectra and the mean of the Montmorillonite spectra, which was also scaled to have unit norm. The abundances of the background spectra at each pixel were chosen by generating values from the beta distribution with parameters $\alpha, \beta = 2, 2$, with different background spectra on the right and left as described above. Each pixel was then scaled so the abundances summed to one. The target pixels were placed at all combinations of x and $y \in \{8, 24, 40, 56, 72, 88, 104, 120\}$. At these pixels the target abundance was set to one and the abundances of the other, background, endmembers were reset to zero.

To generate the actual values at a pixel, the spectrum was constructed by mixing the endmember spectra according to the abundances at the pixel, giving $\mathbf{a}^T\mathbf{S}$. This was scaled by a random value drawn from the Gamma distribution with parameters 4 and 10^5 to get a vector of values $\mathbf{x}_{\text{clean}}$ for the bands. Noise was introduced in each band b at the pixel by using the b-th element of $\mathbf{x}_{\text{clean}}$ as the mean of a Poisson distribution, from which the final value x_b for that pixel and band was drawn.

Admittedly, this is an extreme case, in which the background has two separated modes and the target is between the two modes. In fact it is carefully constructed so that the Gaussian fit to the background will have the target at the center, up to noise, and no background pixels are near this center. All algorithms making the single Gaussian background assumption will fail in this case. And in fact for this data ACE gives an ROC curve essentially along the diagonal, showing random performance, while the MSG-based algorithm has perfect performance. While it might be considered unrealistic, it does demonstrate one way in which the usual Gaussian assumptions can fail, while a multi-subspace algorithms can succeed. In fact MSG has given performance superior to the ACE algorithm on real data, with a caveat: At random times the current MSG algorithm gives wildly wrong behavior, which I believe is a sign of getting stuck in a local minimum. I believe this condition can be detected and fixed, but it requires further work.

4. DISCUSSION

The MSG model proposed here analyzes hyperspectral data by clustering it into a set of subspaces using Gaussian mixture models with constrained covariance matrices. The E-step of the EM algorithm segments the scene into regions with common spectral properties, and presumably few endmembers, while the M-step estimates the subspaces, including the number of dimensions of each. A well-known model-growing algorithm[9] chooses the number of components, and therefore subspaces. The approach avoids potential difficulties in estimating endmembers, as discussed in the introduction and demonstrated in Section 3. In practice it is reasonably fast. In particular a version of the MSG-based material detection algorithm runs in real time on a standard computer.

Clearly, without identifying endmembers the MSG algorithm does not answer all of the questions one might want to pose to the data. Still, the general approach of modeling with multiple subspaces can be used as is for some tasks, and it should be possible to use it as a preliminary stage of analysis. As mentioned above, it segments the image into regions of common spectral characteristics. It may be possible to use one of the current unmixing algorithms to estimate endmembers from the pixels of such a region, with the endmembers constrained to lie in the region's subspace. With the prior segmentation the number of endmembers within a segment is more safely assumed to be small.

For material detection, the example in Section 3.2 was carefully constructed to cause conventional algorithms to fail completely, while MSG gave perfect performance. Real data is unlikely to show such extreme behavior, and in fact the other algorithms such as ACE will often perform better than the version of the MSG algorithm presented here, in part because this MSG-based detection algorithm assumes pure pixels for the target material. However, real data is usually non-Gaussian, and often multi-modal. If the endmember of interest is not well separated from the background endmembers, MSG can perform better, and I have observed MSG out-performing ACE on real data. It is possible that this tends to occur as the number of effective dimensions decreases, since it is harder for the dimensionality to separate background from target. In addition, MSG automatically segments the image, obviating the need for guard regions around the test pixel. In fact MSG behaves correctly when a large fraction of the scene's pixels contain the material of interest, which is quite possible for some applications.

There are a number of areas in which more development of MSG-like algorithms can be useful:

Subspace post-analysis As mentioned above, knowing subspaces that fit a scene should be helpful for estimating endmembers, even if all cannot be reliably estimated. It should also be possible to analyze the relations between the subspaces, e.g., find those which are common, which are contained by another, intersections, etc. Note that any one-dimensional intersections will be endmembers. Algorithms to answer these questions will be complicated by finite sampling and noise, so that subspaces don't quite line up, don't quite intersect, etc.

Local minima The MSG-based material detection algorithm appears to get stuck in local minima some of the time. I believe this can be detected and corrected.

Sub-pixel detection The MSG-based material detection algorithm presented here assumes pure target pixels. While it sometimes detects pixels that are significantly mixed with the background, it should be possible to design an MSG-based algorithm for sub-pixel detection, using subspaces that include background and target endmembers.

Speed While adequate for some applications, the unsupervised model fitting algorithm presented here takes more time than some unmixing algorithms. There are algorithms in the literature for speeding the fitting of Gaussian mixture models, e.g., that of Verbeek et al.,[18] and these should apply directly to MSGs. The large literature on GMMs is a benefit of the MSG model.

Independent of the MSG algorithm, I have argued that the problem of blind unmixing is ill-conditioned on many images, in particular the actual endmembers cannot all be robustly estimated from the data. I suggest an alternative problem that may be well-conditioned: fit the spectra in the image with a set of low-dimensional subspaces. Each pixel then lies near one of these subspaces. Each subspace corresponds to a mixture of a small set of materials, i.e., it is spanned by the corresponding set of endmembers, whether they can be estimated or not. The goal is to thoroughly and efficiently characterize a scene in terms of its subspaces, and extract any endmembers that can be confidently estimated. This capability can then be used as a basis for a variety of applications.

APPENDIX A. M-STEP DERIVATION, ZERO-MEAN

We'll need the inverse and log-determinant of $\mathbf{\Sigma}_c$. To derive these, note that, given $\mathbf{P}_c = \mathbf{V}_c\mathbf{V}_c^T$, the expression $\mathbf{\Sigma}_c = \sigma_{c\|}^2\mathbf{P}_c + \sigma_{c\perp}^2\overline{\mathbf{P}}_c$ is essentially an eigen-decomposition, with eigenvalues $\sigma_{c\|}^2$ and $\sigma_{c\perp}^2$. The inverse can be obtained by inverting the eigenvalues in the decomposition, and the determinant is the product of the eigenvalues, i.e.,

$$\mathbf{\Sigma}_c^{-1} = \sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c, \tag{22}$$

$$\log|\mathbf{\Sigma}_c| = \log\left|\sigma_{c\|}^2\mathbf{P}_c + \sigma_{c\perp}^2\overline{\mathbf{P}}_c\right| \tag{23}$$

$$= M_c \log\sigma_{c\|}^2 + (L - M_c)\log\sigma_{c\perp}^2 \tag{24}$$

where M_c is the multiplicity of the non-zero eigenvalue of $\mathbf{P}_c$, i.e., the dimensionality of the subspace. We will use the last expression later in the derivation, since to begin with we will not assume that we know the M_c's.

Use the symbol Q for the expected log of the complete-data likelihood. We will need to take the derivative of Q with respect to the $\mathbf{P}_c$. To force these to be orthogonal projection matrices add Lagrange constraint terms, which are $\sum_c \mathrm{Tr}\mathbf{\Lambda}_c^T\mathbf{P}_c\overline{\mathbf{P}}_c$ to make the $\mathbf{P}_c$ idempotent, and $\frac{1}{2}\sum_c \mathrm{Tr}\mathbf{\Phi}_c^T\left(\mathbf{P}_c - \mathbf{P}_c^T\right)$ to make them symmetric. With the definitions (4) and (5), we can write Q as

$$Q = \sum_c Z_c \Big\{ \log \Pr(c) - \frac{L}{2}\log(2\pi) - \frac{1}{2}\log\big|\sigma_{c\|}^2 \mathbf{P}_c + \sigma_{c\perp}^2 \overline{\mathbf{P}}_c\big| - \frac{1}{2}\mathrm{Tr}\big[(\sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c)\mathbf{R}_c\big]\Big\}$$
$$+ \sum_c \mathrm{Tr}\boldsymbol{\Lambda}_c^T \mathbf{P}_c \overline{\mathbf{P}}_c + \frac{1}{2}\sum_c \mathrm{Tr}\boldsymbol{\Phi}_c^T\big(\mathbf{P}_c - \mathbf{P}_c^T\big) \qquad (25)$$

The gradient of Q with respect to $\mathbf{P}_c$ is

$$\frac{\partial Q}{\partial \mathbf{P}_c} = -\frac{Z_c}{2}\big\{(\sigma_{c\|}^2 - \sigma_{c\perp}^2)(\sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c) + (\sigma_{c\|}^{-2} - \sigma_{c\perp}^{-2})\mathbf{R}_c\big\} + \boldsymbol{\Lambda}_c\overline{\mathbf{P}}_c - \mathbf{P}_c\boldsymbol{\Lambda}_c + \boldsymbol{\Phi}_c^-, \qquad (26)$$

where $\boldsymbol{\Phi}_c^- = (\boldsymbol{\Phi}_c - \boldsymbol{\Phi}_c^T)/2$, the antisymmetric part of $\boldsymbol{\Phi}_c$. To solve for $\mathbf{P}_c$, set (26) to zero, multiply on the left by $\mathbf{P}_c$ and on the right by $\overline{\mathbf{P}}_c$ to get

$$0 = -\frac{Z_c}{2}(\sigma_{c\|}^{-2} - \sigma_{c\perp}^{-2})\mathbf{P}_c\mathbf{R}_c\overline{\mathbf{P}}_c + \mathbf{P}_c\boldsymbol{\Phi}_c^-\overline{\mathbf{P}}_c. \qquad (27)$$

If we instead multiply on the left by $\overline{\mathbf{P}}_c$ and on the right by $\mathbf{P}_c$, and then take the transpose, we get the same expression, except that the term $\mathbf{P}_c\boldsymbol{\Phi}_c^-\overline{\mathbf{P}}_c$ now has a minus sign because of the antisymmetry of $\boldsymbol{\Phi}_c^-$. Adding the two expressions and discarding the scalar factors, assuming $\sigma_{c\|}^2 \neq \sigma_{c\perp}^2$, gives

$$\mathbf{P}_c\mathbf{R}_c\overline{\mathbf{P}}_c = 0. \qquad (28)$$

Transform to a basis in which $\mathbf{P}_c$ projects onto the subspace of the first M_c basis vectors, and $\overline{\mathbf{P}}_c$ projects onto the complement subspace, spanned by the remaining $L - M_c$ basis vectors. Any square matrix divides into four corresponding blocks, the first M_c rows and the remainder, and the first M_c columns and the remainder. The projection matrices in $\mathbf{P}_c\mathbf{R}_c\overline{\mathbf{P}}_c$ zero everything except the upper-right off-diagonal block. Since $\mathbf{R}_c$ is symmetric, (28) implies that $\mathbf{R}_c$ is block-diagonal in this basis. For this to be the case, $\mathbf{P}_c$ must project onto a subspace spanned by some M_c of $\mathbf{R}_c$'s eigenvectors.

Therefore, to extremize Q with respect to $\mathbf{P}_c$, we choose any subset of eigenvectors of $\mathbf{R}_c$, form the matrix $\mathbf{V}_c$ whose columns are these eigenvectors, and set $\mathbf{P}_c = \mathbf{V}_c\mathbf{V}_c^T$. This is not necessarily a global maximum. Each subset of eigenvectors gives a different extremum, each of which may be a local maximum, inflection point, or local minimum. At the global maximum, however, the subspace should include the eigenvectors with largest eigenvalues, so that the subspace captures as much of the variance as possible.

The remaining question is how many of the principle eigenvectors to include in the subspace. To answer this, we first find the optimal values for $\sigma_{c\|}^2$ and $\sigma_{c\perp}^2$, for a fixed value of M_c. The gradients of Q with respect to the inverse variances are

$$\frac{\partial Q}{\partial \sigma_{c\|}^{-2}} = \frac{Z_c}{2}\big[\sigma_{c\|}^2 \mathrm{Tr}\mathbf{P}_c - \mathrm{Tr}(\mathbf{P}_c\mathbf{R}_c)\big] \qquad \text{and} \qquad (29)$$

$$\frac{\partial Q}{\partial \sigma_{c\|}^{-2}} = \frac{Z_c}{2}\big[\sigma_{c\perp}^2 \mathrm{Tr}\overline{\mathbf{P}}_c - \mathrm{Tr}(\overline{\mathbf{P}}_c\mathbf{R}_c)\big]. \qquad (30)$$

Setting these to zero, the solutions are Equations (7) and (8), since $\mathrm{Tr}(\mathbf{P}_c\mathbf{R}_c)$ is the sum of the M_c largest eigenvalues, $\mathrm{Tr}(\overline{\mathbf{P}}_c\mathbf{R}_c)$ is the sum of the remaining $L - M_c$ eigenvalues, $\mathrm{Tr}\mathbf{P}_c = M_c$, and $\mathrm{Tr}\overline{\mathbf{P}}_c = L - M_c$.

If we insert the solutions for $\mathbf{P}_c$ and the variances into (25), we get Equation (6), up to terms independent of the M_c's. This only depends on M_c and the eigenvalues, so it is straightforward to search for the values of M_c that maximize Q.

APPENDIX B. M-STEP DERIVATION, NON-ZERO MEAN

This derivation will frequently refer to the zero-mean case in Appendix A. The expression for Q is quite similar to that of the zero-mean case (25), except that $\mathbf{R}_c$ is replaced by $\mathbf{R}_c - \boldsymbol{\mu}_c\mathbf{m}_c^T - \mathbf{m}_c\boldsymbol{\mu}_c^T + \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T$, and there is an additional Lagrange constraint term, $\boldsymbol{\lambda}_c^T\overline{\mathbf{P}}_c\boldsymbol{\mu}_c$ to keep the model mean in the subspace. With the definitions (4, 5, 12, 13, 14) we can write Q as

$$\begin{aligned} Q = \sum_c Z_c\Big\{ &\log \Pr(c) - \frac{L}{2}\log 2\pi - \frac{1}{2}\log|\sigma_{c\|}^2\mathbf{P}_c + \sigma_{c\perp}^2\overline{\mathbf{P}}_c| \\ &- \frac{1}{2}\mathrm{Tr}[\sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c](\mathbf{R}_c - \boldsymbol{\mu}_c\mathbf{m}_c^T - \mathbf{m}_c\boldsymbol{\mu}_c^T + \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T)\Big\} \\ &+ \sum_c\big[\mathrm{Tr}\boldsymbol{\Lambda}_c^T\mathbf{P}_c\overline{\mathbf{P}}_c + \mathrm{Tr}\boldsymbol{\Phi}_c(\mathbf{P}_c - \mathbf{P}_c^T) + \boldsymbol{\lambda}_c^T\overline{\mathbf{P}}_c\boldsymbol{\mu}_c\big]. \end{aligned} \tag{31}$$

The gradients with respect to the Gaussian's parameters are

$$\frac{\partial Q}{\partial \boldsymbol{\mu}_c} = Z_c\big[\sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c\big](\mathbf{m}_c - \boldsymbol{\mu}_c) + \sum_c \overline{\mathbf{P}}_c\boldsymbol{\lambda}_c, \tag{32}$$

$$\begin{aligned} \frac{\partial Q}{\partial \mathbf{P}_c} = -\frac{Z_c}{2}\big[&(\sigma_{c\|}^2 - \sigma_{c\perp}^2)(\sigma_{c\|}^{-2}\mathbf{P}_c + \sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c) \\ &+ (\sigma_{c\|}^{-2} - \sigma_{c\perp}^{-2})(\mathbf{R}_c - \boldsymbol{\mu}_c\mathbf{m}_c^T - \mathbf{m}_c\boldsymbol{\mu}_c^T + \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T)\big] \\ &+ \boldsymbol{\Lambda}_c\overline{\mathbf{P}}_c - \mathbf{P}_c\boldsymbol{\Lambda}_c + \boldsymbol{\Phi}_c^- - \boldsymbol{\lambda}_c\boldsymbol{\mu}_c^T, \end{aligned} \tag{33}$$

$$\frac{\partial Q}{\partial \sigma_{c\|}^{-2}} = \frac{Z_c}{2}\big[\sigma_{c\|}^2\mathrm{Tr}\mathbf{P}_c - \mathrm{Tr}\mathbf{P}_c(\mathbf{R}_c - \boldsymbol{\mu}_c\mathbf{m}_c^T - \mathbf{m}_c\boldsymbol{\mu}_c^T + \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T)\big] \qquad \text{and} \tag{34}$$

$$\frac{\partial Q}{\partial \sigma_{c\perp}^{-2}} = \frac{Z_c}{2}[\sigma_{c\perp}^2\mathrm{Tr}\overline{\mathbf{P}}_c - \mathrm{Tr}\overline{\mathbf{P}}_c\mathbf{R}_c]. \tag{35}$$

In addition there are the constraints.

To solve for $\boldsymbol{\mu}_c$, set (32) to zero and use the constraint $\overline{\mathbf{P}}_c\boldsymbol{\mu}_c = 0$ to get

$$\boldsymbol{\mu}_c = \mathbf{P}_c\mathbf{m}_c \qquad \text{and} \tag{36}$$

$$\overline{\mathbf{P}}_c\boldsymbol{\lambda}_c = -Z_c\sigma_{c\perp}^{-2}\overline{\mathbf{P}}_c\mathbf{m}_c. \tag{37}$$

With (36) the term $\mathbf{R}_c - \boldsymbol{\mu}_c\mathbf{m}_c^T - \mathbf{m}_c\boldsymbol{\mu}_c^T + \boldsymbol{\mu}_c\boldsymbol{\mu}_c^T$ becomes $\mathbf{C}_c + \overline{\mathbf{P}}_c\mathbf{m}_c\mathbf{m}_c^T\overline{\mathbf{P}}_c$.

Solving (34) and (35) gives

$$\sigma_{c\|}^2 = \mathrm{Tr}(\mathbf{P}_c\mathbf{C}_c)/\mathrm{Tr}\mathbf{P}_c, \qquad \text{and} \tag{38}$$

$$\sigma_{c\perp}^2 = \mathrm{Tr}(\overline{\mathbf{P}}_c\mathbf{R}_c)/\mathrm{Tr}\overline{\mathbf{P}}_c. \tag{39}$$

To solve for $\mathbf{P}_c$ set (33) to zero and again multiply on the left with $\mathbf{P}_c$ and on the right by $\overline{\mathbf{P}}_c$, and construct a second equation by multiplying on the left with $\overline{\mathbf{P}}_c$ and on the right by $\mathbf{P}_c$, then taking the transpose. The latter is

$$0 = -\frac{Z_c}{2}(\sigma_{c\|}^{-2} - \sigma_{c\perp}^{-2})\mathbf{P}_c\mathbf{C}_c\overline{\mathbf{P}}_c - \mathbf{P}_c\boldsymbol{\Phi}_c^-\overline{\mathbf{P}}_c - \mathbf{P}_c\boldsymbol{\mu}_c\boldsymbol{\lambda}_c\overline{\mathbf{P}}_c \tag{40}$$

while the first is the same except the $\boldsymbol{\Phi}^-$ term has a plus sign, and the last term is absent. Add these to cancel the $\boldsymbol{\Phi}_c^-$ term. Write $\mathbf{P}_c\boldsymbol{\mu}_c\boldsymbol{\lambda}_c\overline{\mathbf{P}}_c$ as $-Z_c\sigma_{c\perp}^{-2}\mathbf{P}_c\mathbf{m}_c\mathbf{m}_c\overline{\mathbf{P}}_c$, using (36) and (37). After discarding an overall factor of Z_c the result is

$$0 = \mathbf{P}_c\big[(\sigma_{c\perp}^{-2} - \sigma_{c\|}^{-2})\mathbf{R}_c + \sigma_{c\|}^{-2}\mathbf{m}_c\mathbf{m}_c^T\big]\overline{\mathbf{P}}_c. \tag{41}$$

Again, $\mathbf{P}_c$ must project onto the span of some eigenvectors of the matrix in square brackets. Unfortunately, the choice of those eigenvectors affects the values of the variance parameters, which then changes the eigenvectors. However, we can iterate between solving for $\mathbf{P}_c$ with the variances fixed, and solving for the variances with the projection matrix fixed. Each of these steps increases the likelihood, so it is monotonically increasing.

ACKNOWLEDGMENTS

This work was partially funded by the United States Naval Air Systems Command under contract No. N68335-10-C-0074.

REFERENCES

[1] Bayliss, J. D., Gualtieri, J. A., and Cromp, R. F., "Analyzing hyperspectral data with independent component analysis," in [*26th AIPR Workshop: Exploiting New Image Sources and Sensors*], 133–143, International Society for Optics and Photonics (1998).

[2] Parra, L., Mueller, K.-R., Spence, C., Ziehe, A., and Sajda, P., "Unmixing hyperspectral data," in [*Advances in Neural Information Processing Systems 12*], 942–948, MIT Press (2000).

[3] Wang, J. and Chang, C.-I., "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *Geoscience and Remote Sensing, IEEE Transactions on* **44**, 1586 – 1600 (June 2006).

[4] Nascimento, J. M. and Dias, J. M. B., "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *Geoscience and Remote Sensing, IEEE Transactions on* **43**(4), 898–910 (2005).

[5] Sajda, P., Du, S., Brown, T. R., Stoyanova, R., Shungu, D. C., Mao, X., and Parra, L. C., "Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain," *IEEE Transaction on Medical Imaging* **23** (Dec. 2004).

[6] Kettig, R. L. and Landgrebe, D. A., "Classification of multispectral image data by extraction and classification of heterogeneous objects," *IEEE Transactions on Geoscience Electronics* **14**, 19–26 (Jan. 1976).

[7] Beaven, S. G., Hazel, G., and Stocker, A. D., "Automated gaussian spectral clustering of hyperspectral data," in [*Proc. SPIE; Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII*], **4725**, 254–266 (2002).

[8] Dempster, A. P., Laird, N. M., and Rubin, D. B., "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)* , 1–38 (1977).

[9] Verbeek, J. J., Vlassis, N., and Kröse, B., "Efficient greedy learning of gaussian mixture models," *Neural computation* **15**(2), 469–485 (2003).

[10] Manolakis, D. G., Lockwood, R., Cooley, T., and Jacobson, J., "Is there a best hyperspectral detection algorithm?," in [*Proc. SPIE 7334, 733402; Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*], (2009).

[11] Harsanyi, J. C. and Chang, C.-I., "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection apporach," *IEEE Transactions on Geoscience and Remote Sensing* **32**, 779–785 (July 1994).

[12] Settle, J., "On the relationship between spectral unmixing and subspace projection," *Geoscience and Remote Sensing, IEEE Transactions on* **34**, 1045 –1046 (July 1996).

[13] "USGS Digital Spectral Library." USGS, 3 October 1997 `ftp://ftpext.cr.usgs.gov/pub/cr/co/denver/speclab/pub/spectral.library/splib04.library/ASCII.LIBRARIES/`. (Accessed: 19 December 2014).

[14] Gerg, I., "matlabHyperspectralToolbox." 20 September 2012 `https://github.com/isaacgerg/matlabHyperspectralToolbox`. (Accessed: 3 February 2016).

[15] Ren, H. and Chang, C.-I., "Automatic spectral target recognition in hyperspectral imagery," *Aerospace and Electronic Systems, IEEE Transactions on* **39**, 1232 – 1249 (Oct. 2003).

[16] Green, A. A., Berman, M., Switzer, P., and Craig, M. D., "A transformation for ordering multispectral data in terms of image quality with implications for noise removal," *IEEE Transactions on Geoscience and Remote Sensing* **26**, 65–74 (1988).

[17] Sajda, P., Du, S., and Parra, L. C., "Recovery of constituent spectra using non-negative matrix factorization," in [*Optical Science and Technology, SPIE's 48th Annual Meeting*], 321–331, International Society for Optics and Photonics (2003).

[18] Verbeek, J. J., Nunnink, J. R., and Vlassis, N., "Accelerated em-based clustering of large data sets," *Data Mining and Knowledge Discovery* **13**(3), 291–307 (2006).