Improving the Hyperspectral Linear Unmixing Problem with Unsupervised Clustering and Covariance Estimates

1 November 2007

Eugene Brevdo
Kim Luu
Air Force Research Laboratory
Presentation Outline

- Hyperspectral imaging background
- Linear mixing model
- Assumptions applied
- Database of materials
- Unsupervised grouping
- Covariance estimates and unmixing
- Some experiments
Hyperspectral Imaging for Satellite Characterization

- Common goals of hyperspectral imaging
  - Identify materials
  - Estimate relative abundances of materials

- Air/Space-based platforms: supervised subpixel classification
  - Each pixel has associated spectral trace
  - Spatial and spectral info in data cube
  - Some “pure” pixels, some mixed

- Ground-based platforms: spectral unmixing
  - Object of interest spans one to few pixels
  - No spatial information available
  - No pure pixels
  - Imperfect representative spectra

Source: Galaxy V, Kris Hamada
Let \( r \) be the observation (column vector), \( \alpha_i \)'s be the abundance values (\( 0 \leq \alpha_i \leq 1 \)), \( m_i \)'s be the “pure pixels”, “pure materials”, or “end-members” that generate the observations.

\[
\begin{pmatrix}
\vdots \\
 r \\
\vdots
\end{pmatrix} = \alpha_1 \begin{pmatrix}
\vdots \\
 m_1 \\
\vdots
\end{pmatrix} + \alpha_2 \begin{pmatrix}
\vdots \\
 m_2 \\
\vdots
\end{pmatrix} + \cdots + \alpha_k \begin{pmatrix}
\vdots \\
 m_k \\
\vdots
\end{pmatrix} + \begin{pmatrix}
\vdots \\
 n \\
\vdots
\end{pmatrix}
\]

where \( n \) is a random vector that represents noise from the observation. Rewritten in matrix notation, we have:

\[
r_{L \times 1} = M_{L \times K} \alpha_{K \times 1} + n_{L \times 1}
\]

where the subscripts denote dimensions, \( M \) is a matrix of the column vectors \( m_k \), and \( \alpha \) is a column vector. For \( T \) observations, we can generalize the mixing process:

\[
R_{L \times T} = M_{L \times K} A_{K \times T} + N_{L \times T}
\]
Under assumptions of Gaussian noise, we can use Least Squared Error optimization to calculate the abundance vector $\alpha$ from the observation $r$. We must know what materials are in the observation in order to accurately estimate $\alpha$.

$$\min_{\alpha} \| M\alpha - r \|^2 = \min_{\alpha} \alpha^T M^T M \alpha - 2r^T M \alpha + r^T r$$

s.t. $\alpha_k \geq 0 \quad k = 1, \ldots, N_M$

$$\sum_{k=1}^{N_M} \alpha_k = 1$$

This problem can be solved using iterative Quadratic Programming algorithms.

Note the columns of $M = [m_1 \ m_2 \ \cdots \ m_K]$ need to be chosen from our database of materials. If we add too many columns to $M$, the error in our abundance estimate generally increases. How do we construct $M$ to get most accurate abundance estimate?
• Sources of variation in spectral data:
  1. Different materials
  2. Different samples of same material
  3. Viewing and Illumination Angle
  4. Space Aging
• Sample of yellow paint on aluminum. Variation in viewing and illumination angle
• Notice specular reflection
Prior Information Assumed

- We know all of the “superclusters” contributing to the observations (e.g., Aluminum, White Paint, Solar Cell).

- But within each supercluster we can have radically different spectra.

Question: How do we go from a set of superclusters to a matrix $M$ for unmixing?
Database Representation

1. Split up superclusters to minimize within-group variation

2. Choose cluster Representatives
   1 trace from each cluster

3. Generate M matrix of 2 or more columns

- First step achieved via Unsupervised clustering – Kernel K-Means
Proposed Solution

Initial problem

Superclusters broken up

Supercluster A

Supercluster B

Cluster B1

Cluster A1

Cluster A2

Cluster A3

Cluster B2
K-Means Unsupervised Clustering

- Database spectra are in 900+ dimensional space.
- Groups points (spectra) into clusters to minimize within-cluster distance and maximize between-cluster distance
- Suboptimal Gaussian expectation maximization estimator
- All operations can be written in terms of inner products on the data points
Kernel K-Means
Orthogonality Kernel

Regular K-Means Grouping brings together neighboring points (least distance to each other)

Orthogonal Kernel K-Means brings together points that are aligned

We can apply this clustering algorithm to each of our “superclusters”
Clusters now “maximally separated” between-group, in some sense.

Need to select or generate traces for inclusion in the $M$ matrix. How to do this?

Two general approaches.
For each cluster, either:

1. Generate a representative
2. Select a member as representative
Centroid Cluster Representatives

Euclidean mean of cluster spectra
Sample Experiment

Simulate Forward Model

For each of 114x114 \((|S_1| \times |S_2|)\) Traces

\[ a_{\text{mix}} = \{10\% \ 25\% \ 50\% \ 75\% \ 90\% \ 100\%\} \]

\[ r = a_{\text{mix}} s_i + (1 - a_{\text{mix}}) s_j \]

Solve for “Yellow Paint on Al” abundance

\[ M = \begin{bmatrix} S_1 & S_2 \end{bmatrix} \]

\[ M = \begin{bmatrix} S_1^{(1)} \cdots S_2^{(k)} \end{bmatrix} \]

\[ \alpha_{\text{yel}} \]

\[ \alpha_{\text{kmean}} = \sum_i \alpha_{\text{yel}}^{(i)} \]
Unmixing error in abundance estimates from spectra of CV 1144-5 (yellow coating)

<table>
<thead>
<tr>
<th>True $a_i$</th>
<th>0%</th>
<th>10%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>90%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error</strong></td>
<td>0.136</td>
<td>0.048</td>
<td>-0.011</td>
<td>-0.008</td>
<td>-0.006</td>
<td>-0.023</td>
<td>-0.046</td>
</tr>
<tr>
<td><strong>Std Dev</strong></td>
<td>0.305</td>
<td>0.309</td>
<td>0.284</td>
<td>0.197</td>
<td>0.124</td>
<td>0.070</td>
<td>0.027</td>
</tr>
<tr>
<td><strong>K-means</strong></td>
<td>0.022</td>
<td>0.014</td>
<td>0.015</td>
<td>-0.003</td>
<td>-0.028</td>
<td>-0.040</td>
<td>-0.032</td>
</tr>
<tr>
<td><strong>Std Dev</strong></td>
<td>0.100</td>
<td>0.127</td>
<td>0.150</td>
<td>0.144</td>
<td>0.139</td>
<td>0.139</td>
<td>0.095</td>
</tr>
<tr>
<td><strong>K-means(4,5)</strong></td>
<td>0.023</td>
<td>0.026</td>
<td>0.025</td>
<td>0.004</td>
<td>-0.024</td>
<td>-0.038</td>
<td>-0.030</td>
</tr>
<tr>
<td><strong>Std Dev</strong></td>
<td>0.100</td>
<td>0.124</td>
<td>0.149</td>
<td>0.142</td>
<td>0.137</td>
<td>0.140</td>
<td>0.095</td>
</tr>
</tbody>
</table>

Conclusions
- Unsupervised clustering of database spectra improves unmixing performance
- Performance stays constant or deteriorates with too many K-means clusters
Using Covariance Information

We now have $M$, but we can take advantage of the within-cluster wavelength covariance information. We minimize

$$\min_{\alpha} \left( M\alpha - r \right)^T A \left( M\alpha - r \right) \quad \text{subject to} \quad \alpha_k \geq 0 \quad k = 1, \ldots, N_M \quad \sum_{k=1}^{N_M} \alpha_k = 1$$

and we see that $c_\nu$ is proportional to an estimate of the within-class covariance for material subclass $\nu$.

It has been shown that $A = C_w^{-1}$ is a good choice, where

$$C_w = \sum_{\nu=1}^{[S]} c_\nu = \sum_{x \in s^{(\nu)}} (x - m_\nu)(x - m_\nu)^T$$

and we see that $c_\nu$ is proportional to an estimate of the within-class covariance for material subclass $\nu$. 

\[ \min_{\alpha} \left\| \tilde{M} \alpha - \tilde{r} \right\|^2 \quad \tilde{M} = A^{1/2} M \quad \tilde{r} = A^{1/2} r \]
Simulate Forward Model

For each of $114 \times 114 \left( |S_1| \times |S_2| \right)$ traces

$a_{\text{mix}} = \{10\% \ 25\% \ 50\% \ 75\% \ 90\% \ 100\% \}$

$r = a_{\text{mix}} s_i + (1 - a_{\text{mix}}) s_j$

Solve for “Yellow Paint on Al” abundance

$M = \begin{bmatrix} \bar{S}_1 & \bar{S}_2 \end{bmatrix}$

$A = I$

$\alpha_{\text{yel}}$

$M = \begin{bmatrix} \bar{S}_1^{(1)} & \cdots & \bar{S}_2^{(k)} \end{bmatrix}$

$A = I$

$\alpha_{\text{yel}} = \sum_i \alpha_{\text{yel}}^{(i)}$

$\alpha_{\text{kmean},\text{cov}} = \sum_i \alpha_{\text{yel},\text{cov}}^{(i)}$
Unmixing error in abundance estimates from spectra of CV 1144-5 (yellow coating)

<table>
<thead>
<tr>
<th>True $a_i$</th>
<th>0%</th>
<th>10%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>90%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Error</td>
<td>0.136</td>
<td>0.048</td>
<td>-0.011</td>
<td>-0.008</td>
<td>-0.006</td>
<td>-0.023</td>
</tr>
<tr>
<td></td>
<td>Std Dev</td>
<td>0.305</td>
<td>0.309</td>
<td>0.284</td>
<td>0.197</td>
<td>0.124</td>
<td>0.070</td>
</tr>
<tr>
<td>Covariance</td>
<td>Error</td>
<td>0.033</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.002</td>
</tr>
<tr>
<td></td>
<td>Std Dev</td>
<td>0.067</td>
<td>0.081</td>
<td>0.073</td>
<td>0.064</td>
<td>0.070</td>
<td>0.077</td>
</tr>
<tr>
<td>K-means</td>
<td>Error</td>
<td>0.016</td>
<td>0.002</td>
<td>-0.001</td>
<td>-0.002</td>
<td>-0.005</td>
<td>-0.008</td>
</tr>
<tr>
<td>Covariance</td>
<td>Std Dev</td>
<td>0.044</td>
<td>0.053</td>
<td>0.050</td>
<td>0.044</td>
<td>0.047</td>
<td>0.051</td>
</tr>
</tbody>
</table>

Conclusions
- Covariance estimates reduce unmixing error
- K-means clustering improves covariance unmixing error to a small degree
Summary

• Clustering the database spectra improves unmixing performance, particularly when we don’t have covariance estimates
• Performance stays constant or deteriorates with too many K-means clusters
• Covariance estimates reduce unmixing error
• K-means clustering improves covariance unmixing error to a small degree