Assignment 4: CS 754

Krushnakant Bhattad	Devansh Jain
190100036	190100044

April 5, 2021

Contents

Question 1

Instructions for running the code:

- After extracting submitted file, look for a directory named `q1`, and `cd` (change directory) to it.
- Run the file `q1.m`.
- The plots can be found in `./plots/`.

Choosing Dictionary and CS-solvers parameters

We know that $\boldsymbol{f_1}$ is sparse in DCT basis and $\boldsymbol{f_2}$ is sparse in Time domain.
However $\boldsymbol{f} = \boldsymbol{f_1} + \boldsymbol{f_2}$ is sparse in neither DCT basis nor Time domain.
So, we create a over-complete dictionary $\boldsymbol{A}$ by concatenating DCT basis and Identity basis (matrix).
$\boldsymbol{f}$ is sparsely representable in $\boldsymbol{A}$.

The OMP implementation provided by Prof. Rajwade as part of Assignment 1 solution (matches with our implementation for Assignment 1 but with additional checks) was used (present in `omp_error.m`).
The parameters to the function are $\boldsymbol{A}$, $\boldsymbol{f}$ and σ.
Line 4 in `omp_error.m` uses σ to calculate error bound $e = n\sigma^2$.

Explaining Code

Creating the Over-complete Dictionary

```
% DCT matrix
dctmat = dctmtx(256);
% Over-complete dictionary for cosine + spikes
A = [dctmat eye(256)];
```

Generating $\boldsymbol{f}$

Here, `s` is the sparsity level of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$.

```
ind1 = randi(256, s, 1);
coeff1 = zeros(256,1);
coeff1(ind1) = rand(s,1)*100;
ind2 = randi(256, s, 1);
coeff2 = zeros(256,1);
coeff2(ind2) = rand(s,1)*100;

f1 = dctmat*coeff1;
f2 = coeff2;
f = f1 + f2;

sigma = 0.01 * abs(mean(f));
f = f + randn(256,1)*sigma;
```

Reconstruction of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$

```
x = omp_error(A, f, sigma);
coeff1_recon = x(1:256);
coeff2_recon = x(257:512);

f1_recon = dctmat*coeff1_recon;
f2_recon = coeff2_recon;
```

Varying σ and fixed sparsity level s

Plots:

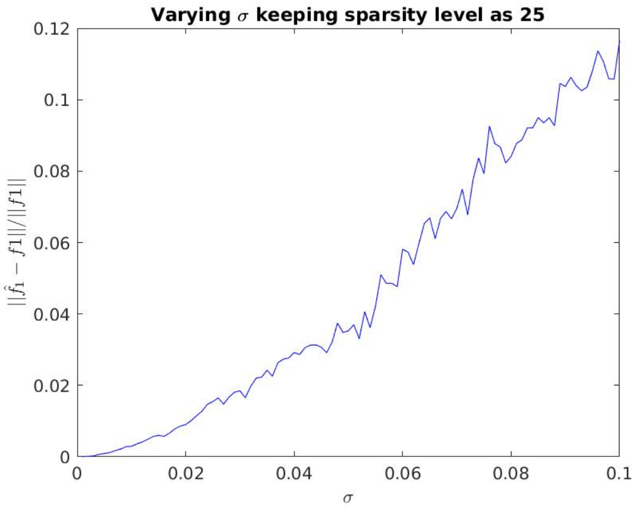


Figure 1: Reconstruction Error in $\boldsymbol{f_1}$

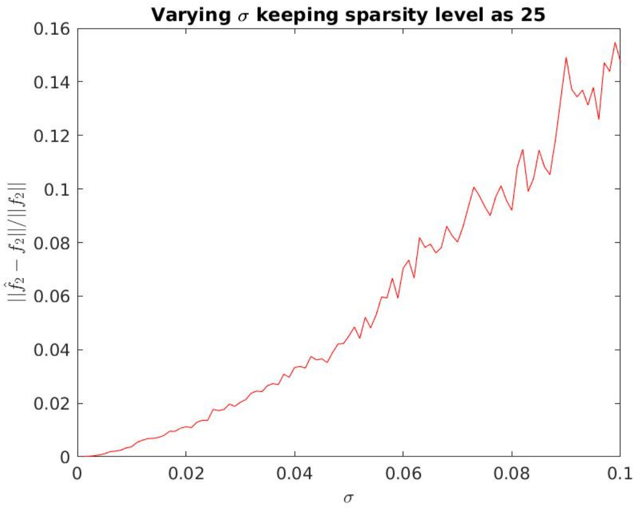


Figure 2: Reconstruction Error in $\boldsymbol{f_2}$

Comments:

- σ ranges from 0.001 to 0.1 × (average value of $\boldsymbol{f_1} + \boldsymbol{f_2}$).
- We can clearly see increase in reconstruction error for both $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ as σ increases.

Varying sparsity level s with fixed σ

Plots:

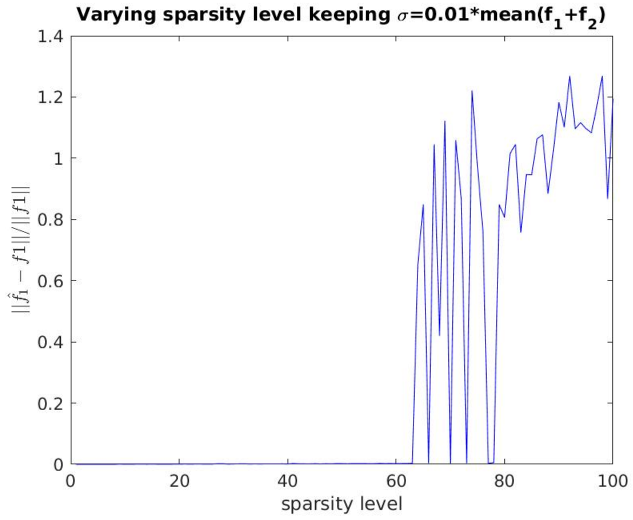


Figure 3: Reconstruction Error in $\boldsymbol{f_1}$

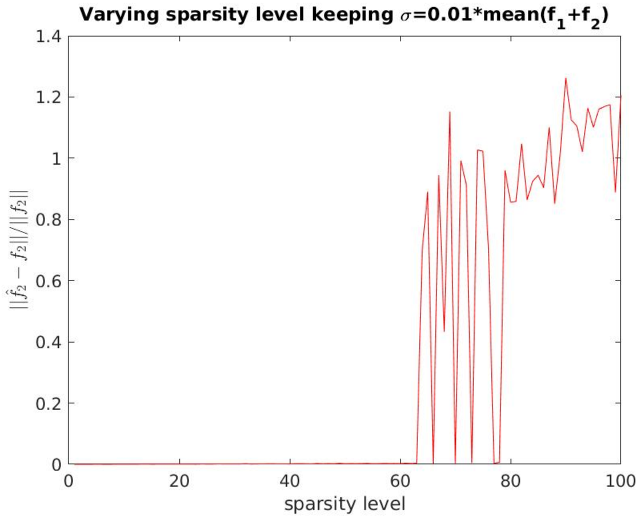


Figure 4: Reconstruction Error in $\boldsymbol{f_2}$

Comments:

- Sparsity level s ranges from 1 to 100.
- Reconstruction error is negligible/acceptable till around sparsity level of about 64 (25% of N=256). We see a sudden rise in error reaching 1 by sparsity level of 70.
- If we zoom into the first part of the plot, we will see a very slow rise (not consistent but visible) in reconstruction error for both $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ with increase in sparsity level s.

Varying magnitude ratio k

Plots:

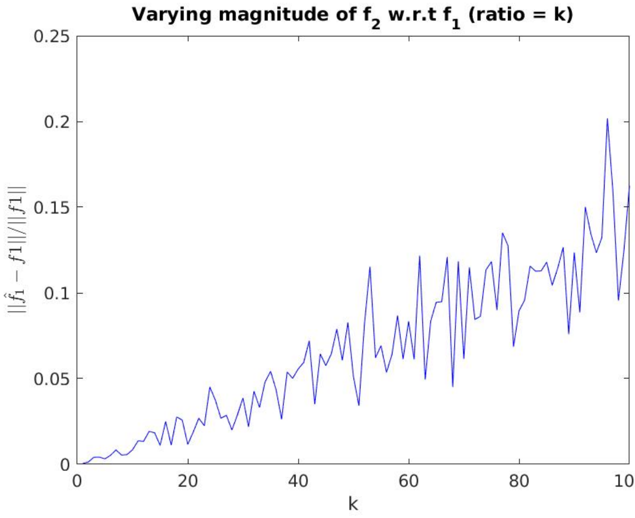


Figure 5: Reconstruction Error in $\boldsymbol{f_1}$

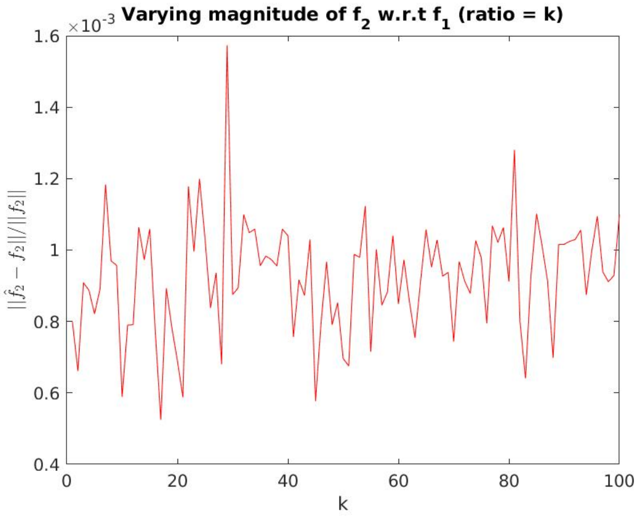


Figure 6: Reconstruction Error in $\boldsymbol{f_2}$

Comments:

- The magnitude of $\boldsymbol{f_2}$ is k times that of $\boldsymbol{f_1}$
- We can clearly see increase in reconstruction error for both $\boldsymbol{f_1}$ as k increases.
  However, the reconstruction error for $\boldsymbol{f_2}$ is almost consistent at around 0.001.
- The possible explanation of this could be the decrease in significance of $\boldsymbol{f_1}$ in $\boldsymbol{f}$, so the error bound in OMP exits when the error value of $\boldsymbol{f_2}$ is less as error value of $\boldsymbol{f_1}$ becomes less significant with increase in k.
- To add to that, σ is based on the average value of $\boldsymbol{f_1} + \boldsymbol{f_2}$, which increases while the average value of $\boldsymbol{f_1}$ remains same, thus giving similar results as the experiment when we were varying σ.
  For $\boldsymbol{f_2}$, increase in σ is counter-acted by the increase in average value of $\boldsymbol{f_2}$, thus giving us almost consistent reconstruction error.

Question 2

Question:

We have studied two greedy algorithms for compressive recovery in class - MP and OMP.

Find out a research paper that proposes a greedy algorithm for CS recovery that is different from OMP and MP. Write down the algorithm in your report, state the key theorem and explain the meaning of the terms involved.

Answer:

There are various greedy algorithms that focus on solving the central problem in CS Recovery,
The Basis Pursuit problem: min $\|\boldsymbol{x}\|_{\ell_1}$ subject to $\boldsymbol{y} = \boldsymbol{\Phi x}$. Here, we'll describe one such algorithm.

The Algorithm: Subspace Pursuit

Paper Title: Subspace Pursuit for Compressive Sensing Signal Reconstruction

Paper by: Wei Dai and Olgica Milenkovic, Department of Electrical and Computer Engineering, UIUC

Link to the paper: `http://arXiv.org/abs/0803.0811v3`

Introduction

The main contribution of this paper is a new algorithm, termed the subspace pursuit (SP) algorithm. It has provable reconstruction capability comparable to that of LP methods, and exhibits the low reconstruction complexity of matching pursuit techniques for very sparse signals. The algorithm can operate both in the noiseless and noisy regime, allowing for exact and approximate signal recovery, respectively. The basic idea behind the SP algorithm is borrowed from coding theory, more precisely, the $A*$ order-statistic algorithm for additive white Gaussian noise channels.

Some Definitions

(Note: Φ^* denotes the transpose of the real valued matrix Φ)

1. *Truncation* and *span* :

Let $\boldsymbol{\Phi} \in \mathbb{R}^{m \times N}$, $\boldsymbol{x} \in \mathbb{R}^N$ and $I \subset \{1, 2, ..., N\}$

$\boldsymbol{\Phi}_I$ denotes the matrix consisting of the columns $\boldsymbol{\Phi}$ of with indices $i \in I$.

$\boldsymbol{x}_I$ is composed of the entries of $\boldsymbol{x}$ indexed by $i \in I$.

$span(\boldsymbol{\Phi}_I)$ denotes the space spanned by the columns of the matrix $\boldsymbol{\Phi}_I$.

2. *Projection* and *Residue* :

Let $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$.
Suppose $\boldsymbol{\Phi}^*\boldsymbol{\Phi}$ is invertible.
Then, the projection of $\boldsymbol{y}$ onto $span(\boldsymbol{\Phi})$ is defined as:

$$\boldsymbol{y}_p = proj(\boldsymbol{y}, \boldsymbol{\Phi}) = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\dagger}\boldsymbol{y}$$

Here, † denotes psuedo-inverse: $\Phi^{\dagger} = \boldsymbol{(\Phi^*\Phi)^{-1}\Phi^*}$

The residue vector of the projection is:

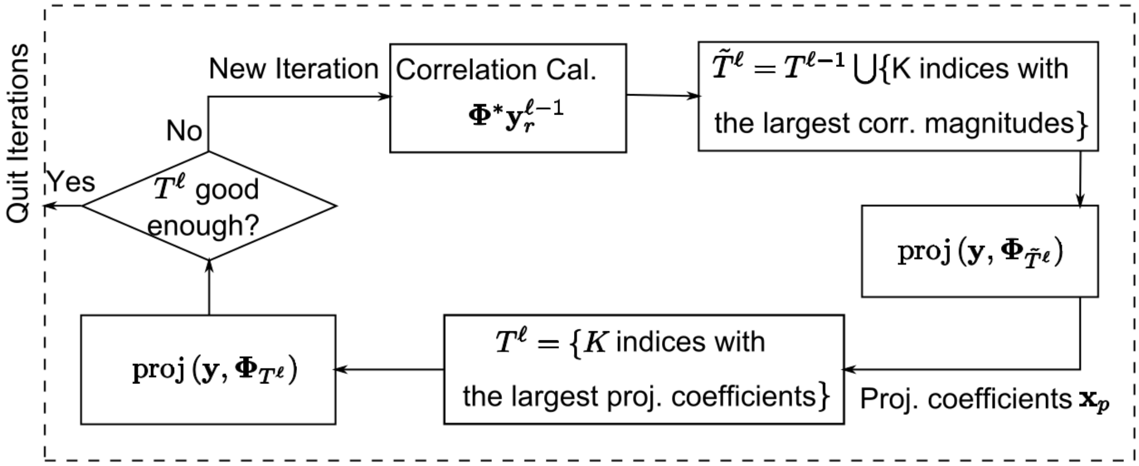
$$\boldsymbol{y}_r = resid(\boldsymbol{y}, \boldsymbol{\Phi}) = \boldsymbol{y} - \boldsymbol{y_p}$$

The Psuedo-Code of the SP Algorithm

```
Input: $K, \boldsymbol{\Phi}, \boldsymbol{y}$
Initialisation:
    1. $T^0 = \{$ $K$ indices corresponding to the largest magnitude entries in the vector $\boldsymbol{\Phi}^* \boldsymbol{y}$ $\}$
    2. $\boldsymbol{y_r^0} = resid(\boldsymbol{y}, \boldsymbol{\Phi}_{T^0})$
Iteration: At the $i^{th}$ iteration, do:
    1. $\tilde{T^i} = T^{i-1} \cup$ $\{$ $K$ indices corresponding to the largest magnitude entries in the vector $\boldsymbol{\Phi}^* \boldsymbol{y_r^{i-1}}$ $\}$
    2. Set $\boldsymbol{x}_p = \boldsymbol{\Phi}^{\dagger}_{\hat{T}^i} \boldsymbol{y}$
    3. $T^i = \{$ $K$ indices corresponding to the largest magnitude entries in the vector $\boldsymbol{x}_p$ $\}$
    4. $\boldsymbol{y_r^i} = resid(\boldsymbol{y}, \boldsymbol{\Phi}_{T^i})$
    5. If $\left\|\boldsymbol{y_r^i}\right\|_2 > \left\|\boldsymbol{y_r^{i-1}}\right\|$, let $T^i = T^{i-1}$ and quit the iteration
Output:
Let $\boldsymbol{x}_{T^i} = \boldsymbol{\Phi}^{\dagger}_{T^i} \boldsymbol{y}$, and set all other entries to 0. Output this estimated signal $\boldsymbol{x}$.
```

Flow Chart

The following Flow Chart can assist in better understanding of the algorithm (here l corresponds to the i in our code- the iterating variable):



Quit Iterations
New Iteration
No
Yes
T^{ℓ} good enough?
Correlation Cal. $\mathbf{\Phi}^* \mathbf{y}_r^{\ell-1}$
$\tilde{T}^{\ell} = T^{\ell-1} \bigcup \{$K indices with the largest corr. magnitudes$\}$
proj $(\mathbf{y}, \mathbf{\Phi}_{\tilde{T}^{\ell}})$
Proj. coefficients $\mathbf{x}_p$
$T^{\ell} = \{K$ indices with the largest proj. coefficients$\}$
proj $(\mathbf{y}, \mathbf{\Phi}_{T^{\ell}})$

(b) Iterations in the proposed Subspace Pursuit Algorithm: a list of K candidates, which is allowed to be updated during the iterations, is maintained.

It is directly taken from the paper, but it does help a great deal in understanding of the algorithm.

Performance Bound Theorems

Non-Noisy Case: Theorem 1

Let $\boldsymbol{x} \in \mathbb{R}^k$ be a K-sparse signal, and let its corresponding measurement be $\boldsymbol{y} = \boldsymbol{\Phi x} \in \mathbb{R}^m$.

If the sampling matrix $\boldsymbol{\Phi}$ satisfies the RIP with constant $\delta_{3K} < 0.165$, then the SP algorithm is guaranteed to **exactly** recover x from y via a finite number of iterations.

Since this is an exact recovery, "bounds" are not needed.

Noisy Case: Theorem 9

Let $\boldsymbol{x} \in \mathbb{R}^k$ be a K-sparse signal, and let its corresponding measurement be $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{e} \in \mathbb{R}^m$, where $\boldsymbol{e}$ denotes the noise vector.

Suppose that the sampling matrix satisfies the RIP with parameter $\delta_{3K} < 0.083$.

Then the reconstruction distortion of the SP algorithm satisfies

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq c_K \|\boldsymbol{e}\|_2$$

Here, $\boldsymbol{x}$ is the original signal, and $\hat{\boldsymbol{x}}$ is the estimated signal.

Also, c_K is a constant independent of $\boldsymbol{x}$, equal to $\dfrac{1 + \delta_{3K} + \delta_{3K}^2}{\delta_{3K}(1 - \delta_{3K})}$

Approximate Case: Theorem 9, Corollary 1

Let $\boldsymbol{x} \in \mathbb{R}^k$ be an approximately K-sparse signal, and let its corresponding measurement be $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{e} \in \mathbb{R}^m$, where $\boldsymbol{e}$ denotes the noise vector.

Suppose that the sampling matrix satisfies the RIP with parameter $\delta_{6K} < 0.083$.

Then the reconstruction distortion of the SP algorithm satisfies

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq c_{2K}(\|\boldsymbol{e}\|_2 + c_K^{'}) \|\boldsymbol{x} - \boldsymbol{x_K}\|_1$$

Repeating notations are same. $\boldsymbol{x_K}$ is the the vector obtained from $\boldsymbol{x}$ by maintaining the K entries with largest magnitude and setting all other entries in the vector to zero.

$c_K^{'}$ is a constant independent of $\boldsymbol{x}$, equal to $\sqrt{\dfrac{1 + \delta_{6K}}{K}}$

Question 3

Given:
Dictionary $\boldsymbol{D}$ contains images $\boldsymbol{d_k}$ ($k \in \{1, \dots, M\}$), each of same dimension as images of class $\mathcal{S}$.
All images $\boldsymbol{s_i}$ in class $\mathcal{S}$ are sparsely represented in dictionary $\boldsymbol{D}$, i.e., $\boldsymbol{s_i} = \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k}$ where $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector.

(a)

Class $\mathcal{S}_1$ which consists of images obtained by applying a known derivative filter to the images in $\mathcal{S}$.
Image $\boldsymbol{x_i}$ in Class $\mathcal{S}_1$ is obtained by applying a known derivative filter $\boldsymbol{d}$ to the image $\boldsymbol{s_i}$ in $\mathcal{S}$.

$$\begin{aligned}
\boldsymbol{x_i} &= \boldsymbol{d} * \boldsymbol{s_i} \\
&= \boldsymbol{d} * \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k} \\
&= \sum_{k=1}^{M} \alpha_k (\boldsymbol{d} * \boldsymbol{d_k}) \quad \text{(Distributive Property of Convolution)}
\end{aligned}$$

As we know that $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector.
All images $\boldsymbol{x_i}$ in class $\mathcal{S}_1$ are sparsely represented in dictionary $\boldsymbol{D_1} = \{\boldsymbol{d} * \boldsymbol{d_k} | \boldsymbol{d_k} \in \boldsymbol{D}\}$.

(b)

Class $\mathcal{S}_2$ which consists of images obtained by rotating a subset of the images in class $\mathcal{S}$ by a known fixed angle α, and the other subset by another known fixed angle β.
Rotating of an image by angle θ is equivalent to pre-multiplying original with a matrix $\boldsymbol{A_\theta}$[1].

$$\begin{aligned}
\boldsymbol{x_i} &= \boldsymbol{A_\theta} \boldsymbol{s_i} \\
&= \boldsymbol{A_\theta} \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k} \\
&= \sum_{k=1}^{M} \alpha_k (\boldsymbol{A_\theta} \boldsymbol{d_k})
\end{aligned}$$

As we know that $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector.
Image $\boldsymbol{x_i}$ in Class $\mathcal{S}_2$ is obtained by rotating the image $\boldsymbol{s_i}$ in $\mathcal{S}$ by either α or β, it can be represented as sparse linear combination of $\boldsymbol{A_\alpha d_k}$ or $\boldsymbol{A_\beta d_k}$.
All images $\boldsymbol{x_i}$ in class $\mathcal{S}_2$ are sparsely represented in dictionary $\boldsymbol{D_2} = \{\boldsymbol{A_\alpha d_k} | \boldsymbol{d_k} \in \boldsymbol{D}\} \cup \{\boldsymbol{A_\beta d_k} | \boldsymbol{d_k} \in \boldsymbol{D}\}$.

[1] refer to comments at the end of question for construction

(c)

Class $\mathcal{S}_3$ which consists of images obtained by applying an intensity transformation $I^i_{new}(x,y) = \alpha(I^i_{old}(x,y))^2 + \beta(I^i_{old}(x,y)) + \gamma$ to the images in $\mathcal{S}$, where α, β, γ are known.
Image $\boldsymbol{x_i}$ in Class $\mathcal{S}_3$ is obtained by by applying an intensity transformation to $\boldsymbol{s_i}$ in $\mathcal{S}$.

$$\begin{aligned}
\boldsymbol{s_i}(x,y) &= \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k}(x,y) \quad (\Longleftrightarrow \boldsymbol{s_i} = \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k}) \\
\boldsymbol{x_i}(x,y) &= \alpha(\boldsymbol{s_i}(x,y))^2 + \beta(\boldsymbol{s_i}(x,y)) + \gamma \\
&= \alpha(\sum_{k=1}^{M} \alpha_k \boldsymbol{d_k}(x,y))^2 + \beta(\sum_{k=1}^{M} \alpha_k \boldsymbol{d_k}(x,y)) + (\gamma) \\
&= (\sum_{k=1}^{M} \alpha\alpha_k^2 (\boldsymbol{d_k}(x,y))^2) + (\sum_{k=1}^{M}\sum_{l>k}^{M} \alpha\alpha_k\alpha_l(\boldsymbol{d_k}(x,y))(\boldsymbol{d_l}(x,y))) + (\sum_{k=1}^{M} \beta\alpha_k(\boldsymbol{d_k}(x,y))) + (\gamma(\mathbf{1})) \\
&(\Longleftrightarrow \boldsymbol{x_i} = (\sum_{k=1}^{M} \alpha\alpha_k^2(\boldsymbol{d_k}. \wedge 2)) + (\sum_{k=1}^{M}\sum_{l>k}^{M} \alpha\alpha_k\alpha_l(\boldsymbol{d_k}. * \boldsymbol{d_l})) + (\sum_{k=1}^{M} \beta\alpha_k \boldsymbol{d_k}) + (\gamma\mathbf{1})
\end{aligned}$$

As we know that $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector, $\{\alpha\alpha_k^2\}_{k=1}^{k=M} \cup \{\alpha\alpha_k\alpha_l\}_{k=1,l>k}^{k=M,l=M} \cup \{\beta\alpha_k\}_{k=1}^{k=M} \cup \{\gamma\}$ is also a sparse vector (percentage sparsity doesn't increase).
All images $\boldsymbol{x_i}$ in class $\mathcal{S}_3$ are sparsely represented in dictionary $\boldsymbol{D_3} = \boldsymbol{D} \cup \{\mathbf{1}, \text{vector contains all 1s}\} \cup \{\boldsymbol{d_k}. \wedge 2 | \boldsymbol{d_k} \in \boldsymbol{D}\} \cup \{\boldsymbol{d_k}. * \boldsymbol{d_l} | \boldsymbol{d_k}, \boldsymbol{d_l} \in \boldsymbol{D}, l > k\}$.

(d)

Class $\mathcal{S}_4$ which consists of images obtained by applying a known blur kernel to the images in $\mathcal{S}$.
Image $\boldsymbol{x_i}$ in Class $\mathcal{S}_4$ is obtained by applying a known blur kernel $\boldsymbol{\omega}$ to the image $\boldsymbol{s_i}$ in $\mathcal{S}$.

$$\begin{aligned}
\boldsymbol{x_i} &= \boldsymbol{\omega} * \boldsymbol{s_i} \\
&= \boldsymbol{\omega} * \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k} \\
&= \sum_{k=1}^{M} \alpha_k (\boldsymbol{\omega} * \boldsymbol{d_k}) \quad \text{(Distributive Property of Convolution)}
\end{aligned}$$

As we know that $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector.
All images $\boldsymbol{x_i}$ in class $\mathcal{S}_4$ are sparsely represented in dictionary $\boldsymbol{D_4} = \{\boldsymbol{\omega} * \boldsymbol{d_k} | \boldsymbol{d_k} \in \boldsymbol{D}\}$.

(e)

Class $\mathcal{S}_5$ which consists of images obtained by applying a blur kernel which is known to be a linear combination of blur kernels belonging to a known set $\mathcal{B}$, to the images in $\mathcal{S}$.
Image $\boldsymbol{x_i}$ in Class $\mathcal{S}_5$ is obtained by applying a blur kernel $\boldsymbol{\omega}$ which is known to be a linear combination[1] of blur kernels belonging to a known set $\mathcal{B}$, to the image $\boldsymbol{s_i}$ in $\mathcal{S}$.

[1] I have assumed that we don't know this linear combination and it may vary for different $\boldsymbol{x_i}$s.

$$
\begin{aligned}
\boldsymbol{\omega} &= \sum_{j=1}^{b} \beta_j \boldsymbol{b_j} \quad (\boldsymbol{b_j} \in \mathcal{B}) \\
\boldsymbol{x_i} &= \boldsymbol{\omega} * \boldsymbol{s_i} \\
&= \boldsymbol{\omega} * \sum_{k=1}^{M} \alpha_k \boldsymbol{d_k} \\
&= \sum_{k=1}^{M} \alpha_k (\boldsymbol{\omega} * \boldsymbol{d_k}) && \text{(Distributive Property of Convolution)} \\
&= \sum_{k=1}^{M} \alpha_k ((\sum_{j=1}^{b} \beta_j \boldsymbol{b_j}) * \boldsymbol{d_k}) \\
&= \sum_{k=1}^{M} \alpha_k (\sum_{j=1}^{b} \beta_j (\boldsymbol{b_j} * \boldsymbol{d_k})) && \text{(Distributive Property of Convolution)} \\
&= \sum_{k=1}^{M} \sum_{j=1}^{b} \alpha_k \beta_j (\boldsymbol{b_j} * \boldsymbol{d_k})
\end{aligned}
$$

As we know that $\{\alpha_k\}_{k=1}^{k=M}$ is a sparse vector, $\{\alpha_k \beta_j\}_{k=1,j=1}^{k=M,j=b}$ is also a sparse vector (percentage sparsity doesn't increase).
All images $\boldsymbol{x_i}$ in class $\mathcal{S}_5$ are sparsely represented in dictionary $\boldsymbol{D_5} = \{\boldsymbol{b_j} * \boldsymbol{d_k} | \boldsymbol{d_k} \in \boldsymbol{D}, \boldsymbol{b_j} \in \mathcal{B}\}$.
Dictionary contains Mb images.

Comments:

- Distributive Property of Convolution has been used in (a), (d) and (e).
  Reference: `http://ccc.inaoep.mx/~a.morales/DSP/pdf/DSP_4_convolution_prop.pdf`, page 36.
- In (c) and (e), number of atoms present in the dictionary has increased significantly, however, percentage sparsity is maintained (in fact, decreased in (c)).
- Derivative filter uses convolution, so does blur kernel, so the results are similar.
  Reference: `https://en.wikipedia.org/wiki/Image_derivatives` (Derivative Filter),
  `https://en.wikipedia.org/wiki/Kernel_(image_processing)` (Blue Kernel)
- Construction of $\boldsymbol{A_\theta}$ used in (b), can be complex and depends on how the image is rotated.
  Several methods are stated here, `https://datagenetics.com/blog/august32013/index.html`.
  In all cases, pixel value in rotated image is some linear combination of original value, this can be represented as a matrix $\boldsymbol{A_\theta}$ of size $n^2 \times n^2$.
  Padding and/or cropping can make the transformation formula more complex.
  In simplest case, $\boldsymbol{x_i}(x, y) = \boldsymbol{s_i}(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$ if $(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$ lies in region else 0.

Question 4

Part (1)

Question:
$\boldsymbol{A}$, a $m \times n$ matrix of rank greater than r , is known apriori.
We seek to minimize $J(\boldsymbol{Q})$:

$$J(\boldsymbol{Q}) = \|\boldsymbol{A} - \boldsymbol{Q}\|_F^2, \text{ where } \boldsymbol{Q} \text{ is a rank-}r \text{ matrix, with } r < m,\ r < n$$

Answer:

The arg min of $J(\boldsymbol{Q})$ is $\boldsymbol{A_r = U\Sigma_r V^T}$
where $\boldsymbol{U\Sigma V^T}$ is the singular value decomposition of $\boldsymbol{A}$
$\boldsymbol{\Sigma_r}$ is same as the matrix $\boldsymbol{\Sigma}$, with all but the top r Singular values in $\boldsymbol{\Sigma}$ taken as 0.

Justification:
A proof of the result above is here: `https://link.springer.com/article/10.1007%2FBF02288367`.
Here, we provide another argument.

For a matrix M let $\sigma_i(M)$ denote the i^{th} largest singular value. WLOG assume $n \geq m$. Then, we have the following:
Lemma: For $m \times n$ matrices X, Y with $q \leq i, j \leq n$, we have

$$\sigma_{i+j-1}(X+Y) \leq \sigma_i(X) + \sigma_j(Y)$$

It is one of Weyl's inequalities.
A proof can be found here: `https://qchu.wordpress.com/2017/03/13/singular-value-decomposition/`, under the section "Additive perturbation (Weyl)".
In this inequality, substitute $X = A - B, Y = B, j = r + 1$ to get:

$$\sigma_{i+r}(A) \leq \sigma_i(A-B) + \sigma_{r+1}(B)$$

Thus, we have,

$$\begin{aligned}\|A - B\|_F^2 &= \sum_{i=1}^{n} \sigma_i^2(A-B) \geq \sum_{i=1}^{n-r} \sigma_i^2(A-B) \\ &\geq \sum_{i=r+1}^{n} \sigma_i^2(A)\end{aligned}$$

And equality is attained when we set $B = A_r$ as defined earlier.

An Intuitive Argument:

Suppose $\boldsymbol{A}$ has rank-p.
Let $\boldsymbol{U\Sigma V^T} = \sum_{i=1}^{p} \sigma_i \boldsymbol{u_i v_i^T}$ be the singular value decomposition of $\boldsymbol{A}$.
For $\boldsymbol{N}$, a $n \times d$ matrix : think of the rows of $\boldsymbol{N}$ as n points in d-dimensional space. The Frobenius norm of $\boldsymbol{N}$ is the square root of the sum of the squared distance of the points to the origin. We will use this interpretation in this problem.

Claim 4.1.1: Interpretation of $\boldsymbol{A_r}$

The rows of $\boldsymbol{A_r}$ are the projections of the rows of $\boldsymbol{A}$ onto the subspace $\boldsymbol{V_r}$ spanned by the first r right singular vectors of $\boldsymbol{A}$.
Proof: Let a be an arbitrary row vector.

Since the $\boldsymbol{v_i}$ are orthonormal, the projection of the vector a onto $\boldsymbol{V_r}$ is given by $\sum_{i=1}^{r}(a \cdot \boldsymbol{v_i})\boldsymbol{v_i}^T$

Thus, the matrix whose rows are the projections of the rows of $\boldsymbol{A}$ onto $\boldsymbol{V_r}$ is given by $\sum_{i=1}^{r} \boldsymbol{A v_i v_i}^T$

This last expression simplifies to $\sum_{i=1}^{r} \boldsymbol{A v_i v_i}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u_i v_i^T} = \boldsymbol{A_r}$
Thus the claim is proved.

Claim 4.1.2: For any matrix B of rank at most r, $\|\boldsymbol{A} - \boldsymbol{A_r}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{B}\|_F^2$

Let $\boldsymbol{B}$ minimize $\|\boldsymbol{A} - \boldsymbol{B}\|_F^2$ among all rank r or less matrices.

Let $\boldsymbol{V}$ be the space spanned by the rows of $\boldsymbol{B}$. The dimension of $\boldsymbol{V}$ is at most k.

Since $\boldsymbol{B}$ minimizes $\|\boldsymbol{A} - \boldsymbol{B}\|_F^2$, it must be that each row of $\boldsymbol{B}$ is the projection of the corresponding row of $\boldsymbol{A}$ onto $\boldsymbol{V}$, otherwise replacing the row of $\boldsymbol{B}$ with the projection of the corresponding row of $\boldsymbol{A}$ onto $\boldsymbol{V}$ does not change $\boldsymbol{V}$ and hence the rank of $\boldsymbol{B}$ but would reduce $\|\boldsymbol{A} - \boldsymbol{B}\|_F^2$
Since each row of $\boldsymbol{B}$ is the projection of the corresponding row of $\boldsymbol{A}$, it follows that $\|\boldsymbol{A} - \boldsymbol{B}\|_F^2$ is the sum of squared distances of rows of $\boldsymbol{A}$ to $\boldsymbol{V}$.

Since $\boldsymbol{A_r}$ minimizes the sum of squared distance of rows of $\boldsymbol{A}$ to any r-dimensional subspace, the claim follows.

Usage:

The best rank-r approximation has an interesting **application to image compression**.

This is based on the fact that, digital images are in fact, stored as matrices; and it is empirically observed that these are inherently low rank.

Thus, we store the image matrix as its low(enough) rank approximation instead.

Suppose $\mathcal{I}$ is an image with values in $\mathbb{R}^{m \times n}$. Then the space needed to store $\mathcal{I}$ is $\mathcal{O}(m * n)$. We store info about the rank$-r$ approximation of $\mathcal{I}$ instead, where $r \ll \min\{m, n\}$.

We perform the SVD of M, and only store the r largest singular values and vectors:

$$\mathcal{I}_{m \times n} \approx \boldsymbol{U_r \Sigma_r V_r^T} = \mathcal{I}_r$$

We store $\boldsymbol{U_r, \Sigma_r, V_r}$ instead of $\mathcal{I}$. The cost is:

$$\text{Cost of } \boldsymbol{U_r} + \text{Cost of } \boldsymbol{\Sigma_r} + \text{Cost of } \boldsymbol{V_r} = \text{Total cost}$$
$$mk + k + nk = k(m + n + 1)$$

This is one magnitude smaller than $\mathcal{O}(m * n)$ when $r \ll \min\{m, n\}$.
The above compression algorithm is otherwise arbitrary. The reason why it works with so promising results, is because of the particular solution to the optimization problem we solved before.

Part (2)

Question:

Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are both known.
We seek to minimize $J(\boldsymbol{R})$ where $\boldsymbol{R}$ is constrained to be orthonormal, and:

$$J(\boldsymbol{R}) = \|\boldsymbol{A} - \boldsymbol{RB}\|_F^2, \text{ where } \boldsymbol{A} \in \mathbb{R}^{n\times m}, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \boldsymbol{R} \in \mathbb{R}^{n\times n}, m > n$$

Answer:

The arg min of $J(\boldsymbol{R})$ with constraint $\boldsymbol{R^T R = I}$, is $\boldsymbol{R = VU^T}$
where $\boldsymbol{USV^T}$ is the singular value decomposition of $\boldsymbol{BA^T}$

Justification:

First, we simplify $J(\boldsymbol{R})$ as follows:

$$\begin{aligned} J(\boldsymbol{R}) &= \|\boldsymbol{A} - \boldsymbol{RB}\|_F^2 \\ &= trace((\boldsymbol{A} - \boldsymbol{RB})^T(\boldsymbol{A} - \boldsymbol{RB})) \\ &= trace(\boldsymbol{A^T A + B^T B - A^T RB - (A^T RB)^T}) \quad \ldots(\text{As } \boldsymbol{R^T R = I}\ ) \\ &= trace(\boldsymbol{A^T A + B^T B}) - 2trace(\boldsymbol{A^T RB}) \quad \ldots(\text{As trace}(\boldsymbol{K^T}) = \text{trace}(\boldsymbol{K})) \end{aligned}$$

Now, $trace(\boldsymbol{A^T A + B^T B})$ is a constant.
Thus to minimize $J(\boldsymbol{R})$ is same as to maximize $trace(\boldsymbol{A^T RB})$ under same constraint.

Further we'll make use of the property that $trace(AB) = trace(BA)$
Thus, firstly, we have $trace(\boldsymbol{A^T RB}) = trace(\boldsymbol{RBA^T})$

Now suppose the singular value decomposition of $\boldsymbol{BA^T}$ is USV^T. Then,

$$trace(\boldsymbol{RBA^T}) = trace(\boldsymbol{RUSV^T}) = trace(\boldsymbol{V^T RUS}) = trace(\boldsymbol{XS}) = \sum_{i=1}^{n} X_{ii} S_{ii}$$

where, $\boldsymbol{X = V^T RU}$ is an orthonormal matrix due to which $|X_{ii}| \leq 1$
The singular values S_{ii} are all non-negative, thus the maximum will be obtained when for all $i, X_{ii} = 1$, i.e. $\boldsymbol{X = I}$.
Thus we need to have $\boldsymbol{V^T RU = I}$, that is $\boldsymbol{R = VU^T}$

Usage:

The given optimization problem is an important and fundamental problem in various fields, like Computer Vision, Computer Graphics, Medical Imaging (especially in a sub-branch called as 'statistical shape analysis') and many more.

In image processing specifically, the solution to above optimization problem is required in quite a few places, of which we'll explain one here.

We'll describe the instance where it is encountered in one of the algorithms in dictionary learning. We use it in the method known as "**Union of Orthonormal bases**"

We represent the signal as: $\boldsymbol{X} = \boldsymbol{AS} + \epsilon$, where $\boldsymbol{X} \in \mathbb{R}^{d\times N}$ is the signal, $\boldsymbol{A} \in \mathbb{R}^{d\times Md}$ is the over-complete dictionary that is a union of ortho-normal bases, $\boldsymbol{S} \in \mathbb{R}^{Md\times N}$ is the sparse co-efficients vector.

$\boldsymbol{A}$ is a the row concatenation of ortho-normal bases A_i for $i \in \{1, 2, ..., M\}$

$\boldsymbol{X}$ is a known signal. Assuming we have fixed bases stored in $\boldsymbol{A}$, the coefficients in $\boldsymbol{X}$ can be estimated using block coordinate descent (BCR) .

Given the coefficients, we next want to update the dictionary.

For all m, we do the following:

1. Get the residual vector: $\boldsymbol{X_m = X} - \sum\limits_{j\neq m} \boldsymbol{A_j S_j}$

2. Solve for $\boldsymbol{A_m}$ as:

$$\boldsymbol{A_m} = \arg\min_{\boldsymbol{A}} \|\boldsymbol{X_m - AS_m}\|^2 \text{ s.t. } \boldsymbol{AA^T = A^T A = I}$$

   Here, $\boldsymbol{X_m} \in \mathbb{R}^{d\times N}$, $\boldsymbol{S_m} \in \mathbb{R}^{d\times N}$ and $\boldsymbol{A} \in \mathbb{R}^{d\times d}$
   It is in this step where we use the minimization problem.

Question 5

Part (a)

Question:

What is hyperspectral unmixing?
You may use an equation to support your answer with symbol meanings carefully explained.

Answer:

Meaning of the Terms

a. Spectral Imaging:

Spectral imaging is imaging that uses multiple bands across the electromagnetic spectrum. While an ordinary camera captures light across three wavelength bands in the visible spectrum, red, green, and blue (RGB), spectral imaging encompasses a wide variety of techniques that go beyond RGB. Spectral imaging may use the infrared, the visible spectrum, the ultraviolet, x-rays, or some combination of the above. It may include the acquisition of image data in visible and non-visible bands simultaneously, illumination from outside the visible range, or the use of optical filters to capture a specific spectral range. It is also possible to capture hundreds of wavelength bands for each pixel in an image.

b. Hyperspectral Imaging:

Hyperspectral imaging, like other spectral imaging, collects and processes information from across the electromagnetic spectrum.[1] The goal of hyperspectral imaging is to obtain the spectrum for each pixel in the image of a scene, with the purpose of finding objects, identifying materials, or detecting processes.
In hyperspectral imaging, a complete spectrum or some spectral information (such as the Doppler shift or Zeeman splitting of a spectral line) is collected at every pixel in an image plane. A hyperspectral camera uses special hardware to capture hundreds of wavelength bands for each pixel, which can be interpreted as a complete spectrum. In other words, the camera has a high spectral resolution.

c. Hyperspectral Unmixing:

Hyperspectral unmixing is a procedure that decomposes the measured pixel spectrum of hyperspectral data into a collection of constituent spectral signatures (or end-members) and a set of corresponding fractional abundances.

Mathematical Formulation

We focus on a relatively simplistic but very representative model, known as the Linear Mixing Model(LMM).

Despite the fact that the LMM is not always true, especially under certain scenarios that exhibit strong non-linearity, it is generally recognized as an acceptable model for many real-world scenarios.

We assume a macroscopic mixing scale in which the incident light interacts with only one material before reflecting off.

Then, the LMM is described as follows:

Let $y_m[n]$ denote the hyper-spectral camera's measurement at spectral band m and at pixel n.

Suppose M is the number of spectral bands, and L is the number of pixels.

Consider the signal $y[n] = [y_1[n], y_2[n], \cdots, y_M[n]]^T \in \mathbb{R}^M$

The LMM is now given by:

$$y[n] = \sum_{i=1}^{n} a_i s_i[n] + v[n] = As[n] + v[n].$$

for $n = 1, 2, \ldots, L$, where:

1. Each $a_i \in \mathbb{R}^M$ for $i = 1, \ldots, N$ is called an endmember signature vector which contains the spectral components of a specific material (indexed by i) in the scene
2. N is the number of endmembers, or materials, in the scene.
3. $A = [a_1 \ldots a_N] \in \mathbb{R}^{M \times N}$ is called the endmember matrix.
4. $s_i[n]$ describes the contribution of material i at pixel n.
   $s[n] = [s_1[n] \ldots s_N[n]] \in \mathbb{R}^N$ is called the abundance vector at pixel n.
5. $v[n] \in \mathbb{R}^M$ simply denotes the noise vector at pixel n.

Some characteristics of the formulation are:

- M is often large— typically more than 200
- The mixing process described by the LMM Equation is a consequence of limited spatial resolution of hyper-spectral cameras. Specifically, one pixel may not be spatially fine enough to contain one material only.
- By nature, the abundance vectors $s[n]$ should satisfy, for every n, $s_i[n] \geq 0$ and $\sum_{i=1}^{n} s_i[n] = 1$

Part (b)

Question:
In equation 40 of the paper, explain how non-negative matrix factorization is used for hyperspectral unmixing.

Answer:
Treat each $y[n]$ for $n = 1, 2, \ldots, L$ as a column vector.
Then these are column concatened to form $\boldsymbol{Y} = [\ y[1]\ y[2]\ \ldots\ y[L]\ ] \in \mathbb{R}^{M \times L}$.
A similar process is carried out on the RHS also, we can express the linear model as:

$$\boldsymbol{Y} = \boldsymbol{AS} + \boldsymbol{V}$$

Here, $\boldsymbol{Y} \in \mathbb{R}^{M \times L}$ is as above, $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is the endmember matrix, $\boldsymbol{S} \in \mathbb{R}^{N \times L}$ is the abundance matrix, and $\boldsymbol{V} \in \mathbb{R}^{M \times L}$ is the noise matrix.

This formulation has special properties:

First, we note that since the end-member matrix $\boldsymbol{A}$ contains the spectral components of specific materials, it is non-negative.

Also, the abundance matrix $\boldsymbol{S}$ has every element in each vector non-negative as explained in the previous part- thus this matrix is non-negative as well.

Thirdly, N, the number of endmembers, or materials, in the scene- is small, $N < M, N < L$.

Moreover the vectors $s[n]$ are empirically(In geoscience and remote sensing, a tremendous amount of effort has been spent on measuring and recording spectral samples of many different materials) observed to be sparse vectors.

Furthermore, the optimization problem we solve here due to the nature of the LMM, is

$$\min_{\boldsymbol{A} \succeq 0, \boldsymbol{S} \succeq \boldsymbol{0}} \|\boldsymbol{Y} - \boldsymbol{AS}\|_F^2$$

This formulation is, in a way, equivalent to the NMF problem, which was:

$$\text{Minimize } E(\boldsymbol{W}, \boldsymbol{H}) = \|\boldsymbol{Y} - \boldsymbol{WH}\| \text{ such that } \boldsymbol{W} \succeq \boldsymbol{0}, \boldsymbol{H} \succeq \boldsymbol{0}$$

Suppose we obtain factors $\boldsymbol{A}$ and $\boldsymbol{S}$ by non-negative matrix factorization(NMF) of $\boldsymbol{Y}$.

Then, we can use $\boldsymbol{A}$ as an estimate of the end-members; and $\boldsymbol{S}$ as the corresponding abundances.

This is what we wanted to estimate: to decompose the measured pixel spectrum of hyperspectral data into a collection of constituent spectral signatures (or end-members) and a set of corresponding fractional abundances.

As we have accomplished this, we have accomplished hyperspectral unmixing. Thus, non-negative matrix factorization can be used for hyperspectral unmixing in the way as described.

Part (c)

Question:

Explain the improvement to non-negative matrix factorization proposed in equation 41 of the paper. (You may explain any two forms each for g and h.)

Answer:

First of all, we impose all the constraints on $\boldsymbol{S}$ that are required:
Let $\mathcal{S} = \{\boldsymbol{S} \mid \forall n: \; s[n] \succeq \boldsymbol{0}; \; \boldsymbol{1}^T s[n] = 1\}$. It is clear that $\boldsymbol{S}$ must belong to $\mathcal{S}$.

Let Q be the set : $Q = \{(A, S) | \boldsymbol{A} \succeq 0, \boldsymbol{S} \in \mathcal{S}\}$. Our optimization is now over Q.

Now we investigate problems with NMF.

First, NMF is NP-hard in general.
For this reason, optimization schemes we see in the current NMF-based blind HU developments are rather pragmatic.
Second, NMF may not guarantee solution uniqueness.
This is a serious issue to the blind HU application, since it means that an NMF solution may not necessarily be the true endmembers and abundances, even in the noiseless case.

In blind HU, NMF is modified to fit the problem better. For this, we use regularizers on $\boldsymbol{A}$ and $\boldsymbol{S}$.

A general formulation is:

$$\min_Q \|\boldsymbol{Y} - \boldsymbol{AS}\|_F^2 + \lambda \cdot g(\boldsymbol{A}) + \mu \cdot h(\boldsymbol{S})$$

Here, g and h are regularizers, and $\lambda, \mu > 0$ are some constants.

Some regularizers are:

1. Abundance regularizer:

   In reality, a given spectral signature is usually composed of a limited number of materials in a hyperspectral scene, and hence the abundance regularization should be selected to be sparsity-prompting. We impose sparsity constraints on the ℓ_1 norm of the columns of $\boldsymbol{S}$.

   We modify the general equation as:

   $g(\boldsymbol{A}) = 0$ and $h(\boldsymbol{S}) = \displaystyle\sum_{i,j} |S_{i,j}|$

2. The $L_{1/2}$ regularizer:

   The $L_{1/2}$ regularizer is an alternative to the L_1 counterpart. The $L_{1/2}$ regularizer is a sparsity-promoting function. Furthermore, the $L_{1/2}$ regularizer not only can provide sparse solutions close to those yielded when L_0 is used but is also computationally efficient.

   We modify the general equation as:

   $g(\boldsymbol{A}) = 0$ and $h(\boldsymbol{S}) = \|S\|_{1/2} = \displaystyle\sum_{k,n=1}^{K,N} \boldsymbol{s}_n(k)^{1/2}$

3. Minimum Volume Regularizer:

   Although we see many choices with the regularizers, the philosophies behind the choices follow a few core principles. For the endmember regularizer g , the principle can be traced back to Minimum Volume in Convex Geometry. A classical example is minimum volume constrained NMF. Here, we modify the general equation as:

   $g(\boldsymbol{A}) = (vol(B))^2$ and $h(\boldsymbol{S}) = 0$

   Here, $vol(B)$ is the simplex volume corresponding to A. (In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions.)

4. Minimum Sum of differences:

   $vol(B)$ is non-convex. Iterated constrained endmember (ICE) and sparsity promoting ICE (SPICE) avoid this issue by replacing $(vol(B))^2$ with a convex surrogate.
   Convex surrogate is a convex function easy to optimize using conventional methods which acts as a proxy for the actual loss we wanted to minimize in the first place.
   One such convex surrogate is the sum of differences between vertices, $g(A) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} ||\boldsymbol{a_i} - \boldsymbol{a_j}||_2^2$