CS 754 - Advanced Image Processing
Assignment 4 - Report

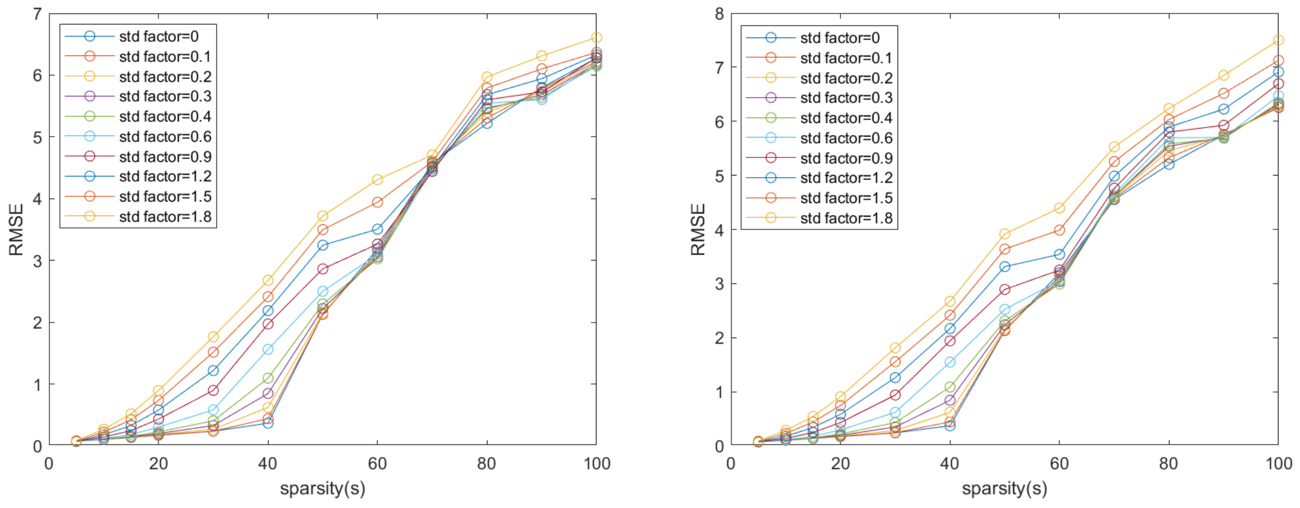
Shaan ul Haque - 180070053
Mantri Krishna Sri Ipsit - 180070032

April 3, 2021

Question 1

1.a and 1.b

We used ISTA to estimate the sparse vector. The final function is sum of cosines, impulses and noise. Thus, the final signal is not sparse in either DCT basis or the temporal basis. In order to overcome this problem, we create an over-complete the dictionary which is concatenation of 256×256 DCT matrix and 256×256 identity matrix column wise. We varied the sparsity level of the signal, in their respective sparse domain, and then for each sparsity level varied the standard deviation of the noise. We calculated maximum Eigenvalue for the dictionary to be used in ISTA algorithm while the hyper-parameter λ was chosen to be 1 as we observed least overall RMSE in the reconstructed signal. Plot for the RMSE error for both the signal f1 and f2 is given below.



(a) $RMSE\ for\ f1$ (b) $RMSE\ for\ f2$

Figure 1: RMSE

We observe that for same sparsity, increasing noise increases the RMSE error while increasing the sparsity level increases the noise as expected from the CS theorems. For large sparsity level(small S) typically in the range of 40 we observe that increasing noise has almost negligible effect on RMSE perhaps suggesting the small dependence of RMSE on noise variance (due to small S).

1.c

We took sparsity level (s) = 20 and noise standard deviation factor = 0.5. We varied k from 0.1 to 10 in small steps initially and then larger steps subsequently. The figure shown below summarizes our result.

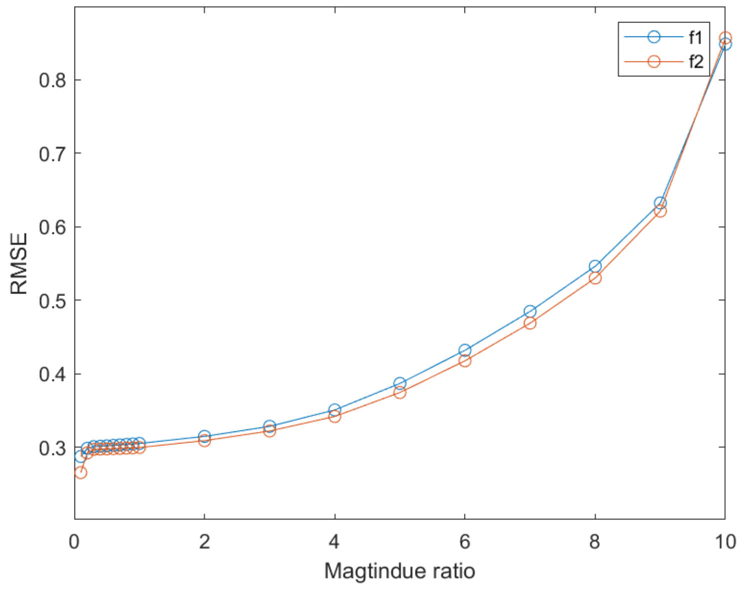


Figure 2: RMSE vs k

We observe that as we increase K the RMSE increases which is as we should expect since noise standard deviation = 0.5(mean(f1)+mean(f2)), thus as magnitude of f2 is increased noise variance increases as well leading to larger RMSE. The hyper-parameter λ was chosen to be 1.75 as we observed least overall RMSE for this value.

Question 2

The algorithm other than OMP and MP presented is called as **Look Ahead Orthogonal Matching Pursuit**. The standard OMP serially adds one component to the support-set that will irreversibly remain in the final estimate, in each iteration. The current choice of the component chosen is independent of the final result. This shortcoming can be overcome using a *look ahead* strategy. In this method, an element is chosen by evaluating its effect on the final performance, in the sense of minimizing the fitting residual at the end of all future iterations. In the current iteration, several potential elements are chosen by the matched filter MF. Each of the potential elements are tested independently, and then one element among them is selected and inducted to the support-set. Among the potential elements, the element that will be selected provides the minimum fitting residual after executing all the future iterations (look ahead strategy). This method is referred to as LAOMP. To develop the LAOMP algorithm, the following algorithm is helpful:

Some notations that were followed:

$$\begin{aligned}
\texttt{resid}(\boldsymbol{y}, \boldsymbol{A}_{\mathcal{T}}) &\triangleq \boldsymbol{y} - \boldsymbol{y}_p, \quad \text{where} \quad \boldsymbol{y}_p = \boldsymbol{A}_{\mathcal{T}} \boldsymbol{A}_{\mathcal{T}}^{\dagger} \boldsymbol{y} \\
\texttt{supp}(\boldsymbol{x}, k) &\triangleq \{\text{the set of indices corresponding to the } k \text{ largest components of } \boldsymbol{x}\} \\
\texttt{supp}(\boldsymbol{x}) &\triangleq \{\text{the set of indices corresponding to all the non-zero components of } \boldsymbol{x}\}
\end{aligned}$$

```
Algorithm 1: look_ahead_resid: Look Ahead Residual
  Input          : $\boldsymbol{A}, \boldsymbol{y}, T$, previous support-set $\mathcal{T}_{\text{old}}$, new index $\tau$
  Assumption     : $i \notin \mathcal{T}_{\text{old}}$, $\hat{\boldsymbol{x}} \in \mathbb{R}^N$
  Initialization:
1 $k = |\mathcal{T}_{\text{old}}| + 1$;
2 $\mathcal{T}_k \leftarrow \mathcal{T}_{\text{old}} \cup \tau$;
3 $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$;
  Iteration      :
4 repeat
5     $k \leftarrow k + 1$;
6     $\tau_k \leftarrow \texttt{supp}(\mathbf{A}^{\intercal}\mathbf{r}_{k-1}, 1)$;
7     $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1} \cup \tau_k$;
8     $\mathbf{r}_k \leftarrow \texttt{resid}(\mathbf{y}, \mathbf{A}_{\mathcal{T}_k})$;
9 until $k{=}T$;
  Output         : $\mathbf{r} \leftarrow \mathbf{r}_k$
```

The above algorithm is the look ahead part of LAOMP. Given the sparsity level T , an intermediate support set $\mathcal{T}$ and a new choice of an atom index τ, the algorithm usually finds a T-element support set at the end and returns the final least squares (LS) residual.
The LAOMP algorithm is based on OMP to construct the support set progressively. In LAOMP, for the choice of a new atom, we take into account its effect on the final performance in the sense of minimizing the residual. We fix an integer parameter λ which is referred to as the look ahead parameter. For the kth iteration, the λ best choices of new atoms are found using MF (Step 3). Then, for each choice of λ atoms, we allow the algorithm to execute until it selects T atoms (Steps 4 and 5. From the λ atoms we choose the one that results in the minimum norm of fitting residual (Step 9). The chosen atom is added to the intermediate support set $\mathcal{T}_{k-1}$ to form the new support set $\mathcal{T}_k$ of cardinality k. The rest of the algorithm remains same as standard OMP. We note that the algorithm must stop when the number of iterations exceeds T . Like most GP algorithms, we assume that the a-priori knowledge of the sparsity level T is available. Also note that, as the value of the look ahead parameter λ increases, better performance can be expected, but at the expense of higher complexity. For $\lambda = 1$, LAOMP will provide the same result as OMP. As this algorithm is based on OMP, the following theorem which presents the performance bounds is also valid for LAOMP. (There is not seperate theorem stating performance bounds was given for LAOMP)

Algorithm 2: Look ahead orthogonal matching pursuit (LAOMP)

```
Input          : $\boldsymbol{A}, \boldsymbol{y}, T$, look ahead parameter $\lambda \leq T$
Define         : $\boldsymbol{n} = [n_1, n_2, \ldots, n_\lambda]^\intercal$ and $\mathcal{T}_\Lambda = [\mathcal{T}_\Lambda(1)], \mathcal{T}_\Lambda(2), \ldots, \mathcal{T}_\Lambda(L)]$
Initialization:
1  $k \leftarrow 0, l \leftarrow 0$;
2  $\boldsymbol{r}_k \leftarrow \boldsymbol{y}, \hat{\boldsymbol{x}}_k \leftarrow \boldsymbol{0}, \mathcal{T}_k \leftarrow \emptyset$;
Iteration      :
3  repeat
4      $k \leftarrow k + 1$;
5      $\mathcal{T}_\Lambda \leftarrow \texttt{supp}(\boldsymbol{A}^\intercal \boldsymbol{r}_{k-1}, \lambda)$;
6      for $l = 1$ to $\lambda$ do
7          $\boldsymbol{rr} \leftarrow \texttt{look\_ahead\_resid}\big(\boldsymbol{y}, \boldsymbol{A}, T, \mathcal{T}_{k-1}, \mathcal{T}_\Lambda(l)\big)$;
8          $n_l \leftarrow \|\boldsymbol{rr}\|_2$
9      end for
10     $l \leftarrow$ Smallest component in $\mathbf{n}$;
11     $\tau_k \leftarrow \mathcal{T}_\Lambda(l)$;
12     $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1} \cup \tau_k$;
13     $\hat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $\hat{\boldsymbol{x}}_{\mathcal{T}_k} = \boldsymbol{A}^\dagger_{\mathcal{T}_k} \boldsymbol{y}$ and $\hat{\boldsymbol{x}}_{\mathcal{T}_k^c} = \boldsymbol{0}$;
14     $\boldsymbol{r}_k \leftarrow \texttt{resid}(\boldsymbol{y}, \boldsymbol{A}_{\mathcal{T}_k})$
15 until $k = T$;
Output         : $\hat{\boldsymbol{x}} \in \mathbb{R}^N$ such that $\hat{\boldsymbol{x}}_{\mathcal{T}_k} = \boldsymbol{A}^\dagger_{\mathcal{T}_k} \boldsymbol{y}$ and $\hat{\boldsymbol{x}}_{\mathcal{T}_k^c} = \boldsymbol{0}$
```

Theorem 1 *Suppose that $\boldsymbol{A}$ satisfies the restricted isometry property (RIP) of order $T+1$ with isometry constant $\delta_{T+1} < \frac{1}{3\sqrt{T}}$. Then, for any $\boldsymbol{x} \in \mathbb{R}^N$ with the number of non-zeros less than or equal to s, LAOMP (OMP) will recover $\boldsymbol{x}$ exactly from $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ in T iterations.*

Theorem 2 (Restricted Isometry Property) *A matrix A is said to obey the restricted isometry property of order N if for any N-sparse vector x*

$$(1-\delta_N)\,\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1+\delta_N)\,\|x\|_2^2$$

where $0 \leq \delta_N \leq 1$ is the smallest positive number such that the above relation holds and it is called "restricted isometry constant" or "isometry constant".

Question 3

We know that by dictionary representation we can compactly express a large number image into small number of dictionary atoms. For any image, each of it's pixel value is the linear combination of those pixels in dictionary atom (when these atoms are represented into images by reshaping the columns). Due to this linearity, any linear operation done on the dictionary atoms gets reflected as it is in the image as well and vice versa. This can be seen from the following equations.
Let an image, $\boldsymbol{I} \in \boldsymbol{S}$, be represented by a linear combination of s elements of a dictionary,

given by $\boldsymbol{D} = \{\boldsymbol{d_1}, \boldsymbol{d_2}...\boldsymbol{d_k}\}$, that is the dictionary has total k atoms.

$$\boldsymbol{I} = \sum_{j=1}^{s} a_j \boldsymbol{d_{m_j}}$$

where a_j are the weights and m_j are the column indices. If the dictionary atoms(columns) are reshaped into matrix, for each pixel in image we can observe that:

$$I(x,y) = \sum_{j=1}^{s} a_j d_{m_j}(x,y) \ \ \forall (x,y) \in \boldsymbol{I}$$

Let us perform any linear operation on the pixels of the image, mathematically:

$$f(I(x_l,y_l)) = \sum_{i=1}^{n} b_i I(x_{li}, y_{li}) \ \ \forall (x_l,y_l) \in \boldsymbol{I}$$

where (x_{li}, y_{li}) are the spatial locations that the linear transformation takes into account (l and i will be related according to the linear transformation, for example $I(x_{li}, y_{li})$ might be the 8-neighborhood pixels around (x_l, y_l), etc). Replacing $I(x_i, y_i)$, we get:

$$f(I(x_l,y_l)) = \sum_{i=1}^{n} b_i \sum_{j=1}^{s} a_j d_{m_j}(x_{li}, y_{li})$$

Interchanging the summation:

$$f(I(x_l,y_l)) = \sum_{j=1}^{s} a_j (\sum_{i=1}^{n} b_i d_{m_j}(x_{li}, y_{li}))$$

The term inside the brackets is the same linear operation albeit on the dictionary atoms. Thus we have shown that:

$$f(\sum_{j=1}^{s} a_j d_{m_j}(x,y)) = f(I(x_l,y_l)) = \sum_{i=1}^{s} a_j f(d_{m_j}(x_l,y_l))$$

3.a

Any derivative filter on an image is a linear function of the pixels. Thus, as shown above, for the images in $\boldsymbol{S_1}$ the dictionary can be easily constructed using the same derivative filter on the atoms of the $\boldsymbol{D}$ as used for the images in $\boldsymbol{S}$. Thus the newly formed dictionary will be $\boldsymbol{D_1} = \{f(\boldsymbol{d_1}), f(\boldsymbol{d_2})...f(\boldsymbol{d_k})\}$, where k is the number of dictionary atoms and f is the derivative filter.

3.b

Again, rotation of any image is linear transformation on the image pixels. But here we need slight tweaking in the sense there are two known angles, α and β, in which the images are rotated. Thus to overcome this issue we will rotate all the atoms (represented as images) by α and β separately and then concatenated both the dictionaries column wise. The intuition behind this is same as we had for Question 1 of this assignment. If $\boldsymbol{D_\alpha} = \{\boldsymbol{d_{1\alpha}}, \boldsymbol{d_{2\alpha}}...\boldsymbol{d_{k\alpha}}\}$ and $\boldsymbol{D_\beta} = \{\boldsymbol{d_{1\beta}}, \boldsymbol{d_{2\beta}}...\boldsymbol{d_{k\beta}}\}$ represent rotated version of the columns (when represented in image form) then the final dictionary would be $\boldsymbol{D_2} = [\boldsymbol{D_\alpha}|\boldsymbol{D_\beta}]$.

3.c

The given transformation is not linear in nature.

$$I_{new}(x,y) = \alpha(I_{old}(x,y))^2 + \beta I_{old}(x,y) + \gamma$$

But we observe some pattern in the transformation which manifests itself in the dictionary as follows. Let the some image $\boldsymbol{I} \in \boldsymbol{S}$ be represented as:

$$\boldsymbol{I} = \sum_{j=1}^{s} a_j \boldsymbol{d_{m_j}}$$

where the symbols have their usual meaning as defined earlier. Writing in pixel form we get:

$$I(x,y) = \sum_{j=1}^{s} a_j d_{m_j}(x,y) \;\; \forall (x,y) \in \boldsymbol{I}$$

Applying the quadratic transformation, we get:

$$I_{new}(x,y) = \alpha(\sum_{j=1}^{s} a_j d_{m_j}(x,y))^2 + \beta(\sum_{j=1}^{s} a_j d_{m_j}(x,y)) + \gamma$$

Opening the square term, we get:

$$I_{new}(x,y) = \alpha(\sum_{j=1}^{s} a_j^2 d_{m_j}^2(x,y) + 2\sum_{i\neq j} a_i a_j d_{m_i}(x,y) d_{m_j}(x,y)) + \beta(\sum_{j=1}^{s} a_j d_{m_j}(x,y)) + \gamma$$

Observing the above term, it is clear that the new dictionary must be concatenation of various functions of the dictionary columns. The new dictionary must have each column squared(element wise), pairwise product(element wise) of each column and a constant ones column(scaling will not matter as it will get absorbed in the coefficient). Thus new dictionary would be concatenation of:

$$\boldsymbol{D_{sq}} = \{\boldsymbol{d_1}^2, \boldsymbol{d_2}^2 ... \boldsymbol{d_k}^2\}$$

where $\boldsymbol{x}^2$ means element wise square of the vector.

$$\boldsymbol{D_{pr}} = \{...\boldsymbol{d_i} \cdot \boldsymbol{d_j}...\} \;\; \forall i \neq j \;\; i,j \in \{1,2,...s\}$$

where $\boldsymbol{x} \cdot \boldsymbol{y}$ means element wise product of the vectors.

$$\boldsymbol{D_c} = ones(\#\ of\ columns(D), 1)$$

Thus, $\boldsymbol{D_3} = [\boldsymbol{D_{sq}} | \boldsymbol{D_{pr}} | \boldsymbol{D} | \boldsymbol{D_c}]$

3.d

Blur kernels are linear transformation on images only, such as the Gaussian blur filter. Thus as done for derivative filter, we can apply this known blur kernel to all the atoms of the dictionary(represented as images) to get the required dictionary. If f is the known blur kernel, then the final dictionary $\boldsymbol{D_4} = \{f(\boldsymbol{d_1}), f(\boldsymbol{d_2}) ... f(\boldsymbol{d_k})\}$.

3.e

Let the set of blur kernel $\boldsymbol{B} = \{\boldsymbol{b_1}, \boldsymbol{b_2} ... \boldsymbol{b_p}\}$. Now let us pick any image $\boldsymbol{I_{blur}}$ from set $\boldsymbol{S_5}$ which is a blurred image of $\boldsymbol{I}$ from $\boldsymbol{S}$.

$$\boldsymbol{I_{blur}} = \sum_{i=1}^{n} c_i \boldsymbol{b_i}(\boldsymbol{I})$$

where c_i is the weights for the blur kernel for the image. From part d, we know that blur kernel gets applied as it is to the atoms of the dictionary as well. Thus, all the new images in the set $\boldsymbol{S_5}$ are essentially linear combination of various blurred atoms from the dictionary with appropriate weights. The new dictionary will be concatenation of various blurred dictionary atoms, $\boldsymbol{D_5} = [\boldsymbol{b_1}(\boldsymbol{D}) | \boldsymbol{b_2}(\boldsymbol{D}) | ... \boldsymbol{b_p}(\boldsymbol{D})]$, where $\boldsymbol{b_i}(\boldsymbol{D}) = \{\boldsymbol{b_i}(\boldsymbol{d_1}), \boldsymbol{b_i}(\boldsymbol{d_2}) ... \boldsymbol{b_i}(\boldsymbol{d_k})\}$, i.e., the blur kernel applied to all the atoms of the dictionary.

Question 4

4.1

$$J(\boldsymbol{A_r}) = \|\boldsymbol{A} - \boldsymbol{A_r}\|_F^2$$

Solving this optimization problem is nothing but finding the best r-rank approximation of the matrix $\boldsymbol{A}$. As we have seen in the lecture, this is done by using the top r-largest singular values and reconstructing $\boldsymbol{A_r}$. This is also known as **Eckart–Young–Mirsky theorem**.

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\intercal \\ \implies \boldsymbol{A_r} &= \boldsymbol{U}(:, 1:r)\boldsymbol{S}(1:r, 1:r)\Big(\boldsymbol{V}(:, 1:r)\Big)^\intercal \end{aligned}$$

This low rank matrix approximation can be used in **Texture Completion**. Generally the texture or any repeating pattern is captured by the low rank approximation. This optimization problem can be used to remove occlusions either from a photo or a video and do a robust repair, which is an application of **Robust PCA**.

4.2

$$J(\boldsymbol{R}) = \|\boldsymbol{A} - \boldsymbol{R}\boldsymbol{B}\|_F^2$$

Please turn over

where $\boldsymbol{R}$ is an orthonormal matrix i.e, $\boldsymbol{R}\boldsymbol{R}^\intercal = \boldsymbol{R}^\intercal\boldsymbol{R} = \boldsymbol{I}$. This optimization is solved as follows:

$$\begin{aligned}
\min \|\boldsymbol{A} - \boldsymbol{R}\boldsymbol{B}\|_F^2 &= \min \operatorname{Tr}\Big((\boldsymbol{A} - \boldsymbol{R}\boldsymbol{B})^\intercal (\boldsymbol{A} - \boldsymbol{R}\boldsymbol{B}) \Big) \\
&= \min \operatorname{Tr}\Big((\boldsymbol{A}^\intercal - \boldsymbol{B}^\intercal\boldsymbol{R}^\intercal)(\boldsymbol{A} - \boldsymbol{R}\boldsymbol{B}) \Big) \\
&= \min \operatorname{Tr}\Big(\boldsymbol{A}^\intercal\boldsymbol{A} - 2\boldsymbol{A}^\intercal\boldsymbol{R}\boldsymbol{B} + \boldsymbol{B}^\intercal\boldsymbol{B} \Big) \\
&= \max \operatorname{Tr}\Big(\boldsymbol{A}^\intercal\boldsymbol{R}\boldsymbol{B} \Big) \\
&= \max \operatorname{Tr}\Big(\boldsymbol{R}\boldsymbol{B}\boldsymbol{A}^\intercal \Big) \qquad \text{since trace(PQ) = trace(QP)} \\
\text{using SVD on } \boldsymbol{B}\boldsymbol{A}^\intercal \quad \boldsymbol{B}\boldsymbol{A}^\intercal &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\intercal \\
&= \max \operatorname{Tr}\Big(\boldsymbol{R}\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\intercal \Big) \\
&= \max \operatorname{Tr}\Big(\boldsymbol{V}^\intercal\boldsymbol{R}\boldsymbol{U}\boldsymbol{D} \Big) \\
&= \max \operatorname{Tr}\Big(\boldsymbol{Z}(\boldsymbol{R})\boldsymbol{D} \Big) \\
&= \max \sum_i z_{ii} d_{ii} \le \sum_i d_{ii} \quad (\because \boldsymbol{Z}(\boldsymbol{R})^\intercal \boldsymbol{Z}(\boldsymbol{R}) = \boldsymbol{I})
\end{aligned}$$

The maximum is achieved for $\boldsymbol{Z}(\boldsymbol{R}) = \boldsymbol{I}$, i.e.,

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^\intercal$$

This algorithm is used in the **Tomography under unknown angles for 3D images**, where we try to estimate the rotation matrix $\boldsymbol{R}$. This problem is also known as **Orthogonal Procrustes** problem.

Question 5

5.a

The problem of hyperspectral unmixing is defined using the following constrained convex optimization problem:

$$\hat{\boldsymbol{s}}[n] = \arg\min_{\boldsymbol{s}[n]\in\mathcal{S}} \|\boldsymbol{y}[n] - \boldsymbol{A}\boldsymbol{s}[n]\|_2^2, \text{ such that } \sum_{i=1}^{N} s_i[n] = 1, s_i[n] \ge 0, i = 1, \dots, N,\ n = 1, \dots, L$$

where

- $\boldsymbol{y}[n] = [y_1[n], y_2[n], \dots, y_M[n]]^\intercal \in \mathbb{R}^M$ is a vector of hyperspectral camera measurements where $y_m[n]$ denotes the camera's measurement at spectral band m and pixel n. There are a total of M spectral bands and L pixels.

- $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_N] \in \mathbb{R}^{M \times N}$ is called as the *endmember matrix* and each $\boldsymbol{a_i}$ is called *endmember signature vector* where N is the total number of endmembers or the materials in the scene.
- $\boldsymbol{s}[n] = [s_1[n], \dots, s_N[n]]^\intercal \in \mathbb{R}^N$ is called the *abundance vector* at pixel n where $s_i[n]$ describes the contribution of material i at pixel n.

5.b

We have the block model

$$\boldsymbol{Y} = \boldsymbol{AS}, \boldsymbol{S} \in \mathbb{R}^{K \times L}, \boldsymbol{Y} \in \mathbb{R}^{M \times L}$$

If S denotes the set of indices of active materials in the measured data $\boldsymbol{Y}$, and if $\boldsymbol{A}_S$ denotes the matrix made from the columns of $\boldsymbol{A}$ indexed by S. If the set of true abundance maps $\{\boldsymbol{s}^i\}_{i \in S}$ are assumed to be linearly independent, then $\mathcal{R}(\boldsymbol{Y}) = \mathcal{R}(\boldsymbol{A}_S)$, where $\mathcal{R}$ denotes the range space of its argument. This gives us equation (40):

$$\|\boldsymbol{S}\|_{\text{row}-0} < \text{spark}(\boldsymbol{A}) - 1$$

Here spark($\boldsymbol{A}$) denotes the smallest number of linearly dependent columns of $\boldsymbol{A}$ and $\|\boldsymbol{S}\|_{\text{row}-0}$ denotes the number of non-zero rows in $\boldsymbol{S}$. So, the problem requirements that we have are stated as follows:

- $\boldsymbol{A} \geq 0$, $\boldsymbol{S} \geq 0$ (based on physical constraints) and $\boldsymbol{Y} = \boldsymbol{AS}$.
- As $\mathcal{R}(\boldsymbol{Y}) = \mathcal{R}(\boldsymbol{A}_S)$, we are trying to represent the columns of $\boldsymbol{Y}$ in the basis formed using columns of $\boldsymbol{A}$.
- As per the inequality mentioned above, we have $K < M$

All the above requirements are captured by performing NMF on $\boldsymbol{Y}$.

5.c

Standard 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 be necessarily be the true endmembers and abundances, even in the noiseless case. Hence, for the case of blind HU, NMF is modified to fit the problem better. The unified NMF-based blind HU problem can be formulated as follows:

$$\min_{\boldsymbol{A} \geq 0, \boldsymbol{S} \in \mathcal{S}^L} \|\boldsymbol{Y} - \boldsymbol{AS}\|_F^2 + \lambda \cdot g(\boldsymbol{A}) + \mu \cdot h(\boldsymbol{S})$$

where $\mathcal{S}^L = \{\boldsymbol{S} | \boldsymbol{s}[n] \geq \boldsymbol{0}, \boldsymbol{1}^\intercal \boldsymbol{s}[n] = 1, 1 \leq n \leq L\}$, g and h are regularizers and λ and μ are positive constants. The following two variations of blind HU are explained:

1. **Iterated Constrained Endmember**(ICE):
   Here, $h(\boldsymbol{S}) = 0$ and

   $$g(\boldsymbol{A}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|\boldsymbol{a}_i - \boldsymbol{a}_j\|_2^2$$

   which is the sum of differences between the vertices.

2. **Sparsity Promoting Iterated Constrained Endmember**(SPICE):

$$g(\boldsymbol{A}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|\boldsymbol{a}_i - \boldsymbol{a}_j\|_2^2$$

$$h(\boldsymbol{S}) = \sum_{i=1}^{N} \gamma_i \left\|\boldsymbol{s}^i\right\|_1$$

Here, we promote sparsity on the columns of $\boldsymbol{S}$