Rank Minimization for Blind Deconvolution

Anh-Huy Phan

December 19

Convolutive Mixing Model I

We consider a mixing model

$$\boldsymbol{y} = \boldsymbol{w} * \boldsymbol{x} + \boldsymbol{n}$$

where

- $\boldsymbol{x}$ is the unknown source signal
- $\boldsymbol{w}$ is the convolutive kernel,
- "*" denotes the convolution

Problem

Retrieve the signal $\boldsymbol{s}$ from the mixtures $\boldsymbol{y}$

The blind deconvolution problem is fundamental in many applications, ill-posed can have infinite number of solutions.

Lifting Method for single channel I

Assume that the signal has length L and $\boldsymbol{w}$ lies in a fixed subspace spanned by $\mathbf{B}$.

$$\boldsymbol{w} = \mathbf{B}\boldsymbol{h} \tag{1}$$

$$\boldsymbol{x} = \mathbf{C}\boldsymbol{l} \tag{2}$$

where $\mathbf{B} \in \mathbb{R}^{L\times K}$, $\boldsymbol{h} \in \mathbb{R}^{K}$, $\mathbf{C} \in \mathbb{R}^{L\times N}$ and $\boldsymbol{l} \in \mathbb{R}^{N}$.
$\mathbf{B}$ can be an identity matrix.

Lifting Method for single channel II

Expanding $\boldsymbol{x}$ and $\boldsymbol{w}$, we get the linear combinations:

$$
\begin{aligned}
\boldsymbol{y} &= l_1 \boldsymbol{w} * \mathbf{C}_1 + \cdots + l_N \boldsymbol{w} * \mathbf{C}_N \\
&= [circ(\mathbf{C}_1)\mathbf{B}, \cdots, circ(\mathbf{C}_N)\mathbf{B}] \begin{bmatrix} l_1 \boldsymbol{h} \\ \vdots \\ l_N \boldsymbol{h} \end{bmatrix} \qquad (3) \\
&= \mathbf{A}\,\mathrm{vec}\left(\boldsymbol{h}\boldsymbol{l}^T\right) \qquad (4)
\end{aligned}
$$

where $\mathbf{A}$ is the lifting linear operator and $circ(\mathbf{C}_i)$ represents circulant matrix

$$
circ(\mathbf{C}_i) = \begin{bmatrix} C_{1i} & C_{2i} & \ldots & C_{Li} \\ C_{Li} & C_{1i} & \ldots & C_{L-1i} \\ \vdots & \ddots & & \vdots \\ C_{2i} & C_{3i} & \ldots & C_{1i} \end{bmatrix}. \qquad (5)
$$

Lifting Method for single channel III

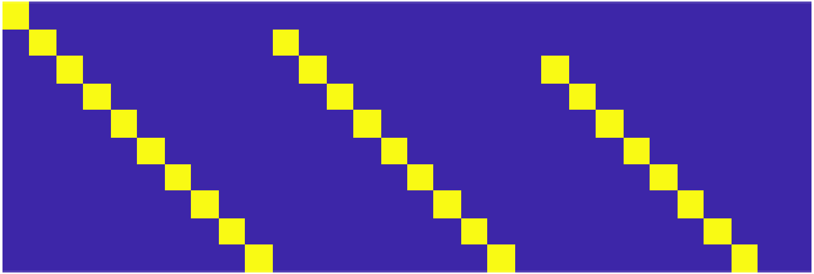


Figure: A lifting matrix for convolutive model with kernel length 3 and signal length 10. There is no fixed basis $\mathbf{B}$ for the kernel, $\boldsymbol{h}$. The lifting matrix is sparse.

Rank Minimization in Lifting Method I

Blind deconvolution problem is recast as finding a rank-1 matrix $\mathbf{X} = \boldsymbol{x}\boldsymbol{h}^T$ which holds

$$\boldsymbol{y} = \mathbf{A}\,\mathrm{vec}(\mathbf{X})$$

or finding a matrix $\mathbf{X}$ with minimal rank

$$\min \quad \|\mathbf{X}\|_* \tag{6}$$

$$\text{s.t.} \quad \boldsymbol{y} = \mathbf{A}\,\mathrm{vec}(\mathbf{X}) \tag{7}$$

The nuclear norm is used in place of the rank function.

```
X = cp.Variable(szX)
objective = cp.Minimize(cp.norm(X, 'nuc'))
constraints = [ALift @ cp.reshape(X,(szX[0]*szX[1],1)) == y.reshape(-1,1)]
prob = cp.Problem(objective, constraints)
prob.solve()
```

Rank Minimization in Lifting Method II

Can the rank minimization algorithm extract the original signal?

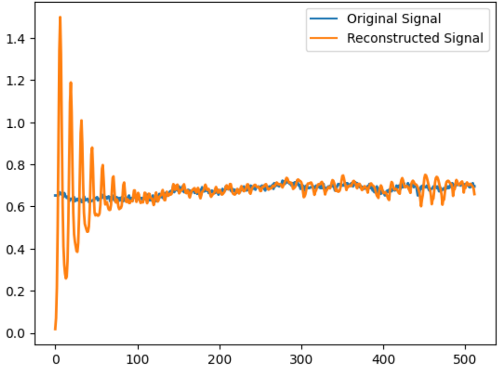


Figure: An illustration of the rank minimization algorithm for blind deconvolution.

- The signal $\boldsymbol{x}$ and the filter kernel $\boldsymbol{h}$ can be extracted from the best rank-1 approximation of the optimal $\mathbf{X}^\star \approx \boldsymbol{x}\boldsymbol{h}^T$.
- The extracted signal can capture the main waveform of the source, but exhibits undesired oscillations.
- *Rank constraint is not sufficient*
- Need more ingredients to fix the algorithm.

Smoothness Constraint I

$$\begin{aligned} \min \quad & \|\mathbf{X}\|_* + \gamma \sum_{l=1}^{L} \|\mathbf{X}(:,l)\|_{TV} \\ \text{s.t.} \quad & \boldsymbol{y} = \mathbf{A}\,\mathrm{vec}(\mathbf{X}) \end{aligned}$$

```
gamma = 0.1
Xsm = cp.Variable(szX)
g = 0
for k in range(len(h)):
  g = g + cp.norm(Xsm[1:,k]-Zs[:-1,k],1)

objective = cp.Minimize(cp.norm(Xsm, 'nuc') + gamma*g)
constraints = [ALift @ cp.reshape(Xsm,(szX[0]*szX[1],1)) == y.reshape(-1,1)]
prob = cp.Problem(objective, constraints)
prob.solve()
```

Smoothness Constraint II

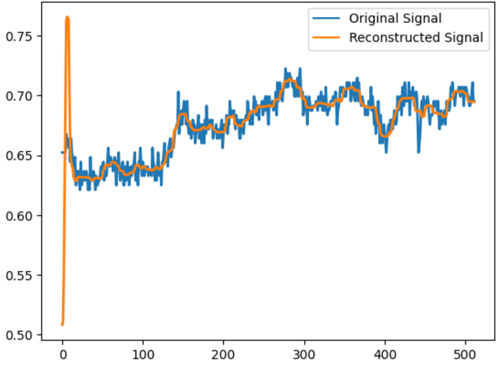


Figure: An illustration of the rank minimization + smoothness constraint for blind deconvolution.

- Rank+ smoothness minimization algorithm successfully retrieves the signal.
- CVX and CVXPY are slow for long signals
- Need a more practical algorithm

ADMM I

►

$$\begin{aligned} min \quad & \|\mathbf{Z}\|_* + \gamma\|\mathbf{X}\|_{TV} + i_A(\mathbf{F}) \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{F} \\ & \mathbf{X} = \mathbf{F} \end{aligned}$$

where $i_A(\mathbf{F})$ is the indicator for the convex set of $\mathbf{F}$ which hold $\boldsymbol{y} = \mathbf{A}vec(\mathbf{F})$

ADMM II

- ADMM updates

$$\begin{aligned}
\mathbf{Z}^{(k+1)} &= \arg\min \|\mathbf{Z}\|_* + \frac{\lambda}{2}\|\mathbf{Z} - \mathbf{F}^{(k)} - \mathbf{T}_z^{(k)}\|_F^2 \\
&= soft_value_thresholding(\mathbf{F}^{(k)} + \mathbf{T}_z^{(k)}, 1/lambda) \\
&= \mathbf{U}\,\mathrm{diag}(\max(\boldsymbol{\sigma} - \frac{1}{\lambda}, 0))\mathbf{V}^T
\end{aligned}$$

$$\begin{aligned}
\mathbf{X}^{(k+1)} &= \arg\min \gamma\|\mathbf{X}\|_{TV} + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{F}^{(k)} - \mathbf{T}_x^{(k)}\|_F^2 \\
&= \mathrm{prox}_{TV}(\mathbf{F}^{(k)} + \mathbf{T}_x^{(k)}, \gamma/\lambda)
\end{aligned}$$

ADMM III

$$
\begin{aligned}
\mathbf{F}^{(k+1)} &= \arg\min i_A(\mathbf{F}) + \frac{\lambda}{2}(\|\mathbf{X}^{(k+1)} - \mathbf{T}_x^{(k)} - \mathbf{F}^{(k)}\|_F^2 + \|\mathbf{Z}^{(k+1)} - \mathbf{T}_z^{(k)} - \mathbf{F}^{(k)}\|_F^2) \\
&= \arg\min \quad \|\mathbf{F} - \mathbf{D}\|_F^2 \\
&\quad \text{s.t.} \quad \boldsymbol{y} = \mathbf{A}\,\mathrm{vec}(\mathbf{F})
\end{aligned}
$$

where $\mathbf{D} = \frac{1}{2}(\mathbf{X}^{(k+1)} - \mathbf{T}_x^{(k)} + \mathbf{Z}^{(k+1)} - \mathbf{T}_z^{(k)})$
Lagrangian for the sub-problem which updates $\mathbf{F}$

$$
L(\boldsymbol{f}, \nu) = \frac{1}{2}\|\boldsymbol{f} - \boldsymbol{d}\|_F^2 + \nu^T(\boldsymbol{y} - \mathbf{A}\boldsymbol{f})
$$

ADMM IV

Set the gradient of $L(\boldsymbol{f}, \nu)$ to zero

$$\frac{\partial L}{\partial \boldsymbol{f}} = \boldsymbol{f} - \boldsymbol{d} - \mathbf{A}^T \nu = 0$$

to get the optimal

$$\boldsymbol{f}^\star = \mathbf{A}^T \nu + \boldsymbol{d}$$

and the dual problem is given by

$$g(\nu) = -\frac{1}{2}\boldsymbol{\nu}^T \mathbf{A}\mathbf{A}^T \boldsymbol{\nu} + \nu^T \boldsymbol{y}$$

which has the optimal dual variable

$$\boldsymbol{u}^\star = (\mathbf{A}\mathbf{A}^T)^{-1} \boldsymbol{y}$$

ADMM V

How to compute $\mathbf{A}\mathbf{A}^T$ and its inverse?

```python
f = functions.norm_tv(tol=10e-7, dim=1,verbosity = 0)

for kiter in range(100):
  # update Z
  D1 = Fk + T1
  ud,sd,vd = np.linalg.svd(D1, full_matrices=False)
  sd = np.maximum(sd-1/lambda_,0)
  Zk = ud @ np.diag(sd) @ vd

  # Update X
  # sol = argmin_{z} 0.5*||x - z||_2^2 + gamma * ||x||_TV
  D2 = Fk + T2
  param = {'tol': 1e-7, 'verbose': 0}
```

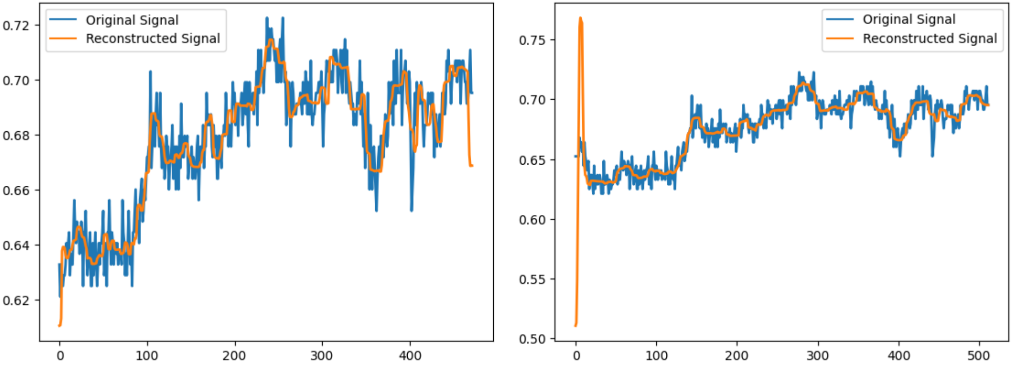
ADMM VI

```python
Xk = np.zeros(szX)

for k in range(szX[1]):
  solk = f.prox(D2[:,k],gamma/lambda_)
  Xk[:,k] = solk

# Update F
K2 = Xk-T2
K1 = Zk-T1
K = (K1+K2)/2
yK = y - ALift @ K.T.reshape(-1)
dualF = yK / szX[1]
dualF[:szX[1]] = yK[:szX[1]] / np.arange(1, szX[1]+1)

Fk = ALift.T@dualF + K.T.reshape(-1)

#Fk = ATmap(dualF) + K
```

ADMM VII

```python
  Fk = Fk.reshape([szX[1], szX[0]]).T

  # Update dual
  T1 = T1 + Fk - Zk
  T2 = T2 + Fk - Xk

  #
  errZF = np.linalg.norm(Zk-Fk,'fro')
  errXF = np.linalg.norm(Xk-Fk,'fro')
  errXZ = np.linalg.norm(Xk-Zk,'fro')

  print(f'{kiter} | d(Z,F)= {errZF} | d(X,F)= {errXF} | d(X,Z) = {errXZ}')

  err.append([errZF,errXF,errXZ])

  if np.sum([errZF,errXF,errXZ])/3 < 1e-4:
    break
```

ADMM VIII



(a) $\gamma = 0.01$

(b) $\gamma = 0.1$

Figure: An illustration of ADMM for blind deconvolution.