ADMM for Structured Optimization Problems: Formulations and Applications

Yassine Nabou
Ph.D. Student

University Politehnica Bucharest, Romania
Faculty of Automatic Control and Computer science

October 2021

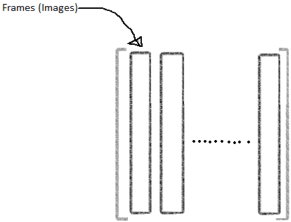
Outline

Video: Sequence of frames (images)

Motivation

- We often need to identify activities that stand out from the background.
- In order for a car to decide what to do next: accelerate, apply brakes or turn, it needs to know where all the objects are around the car and what those objects are.

Stack the video frames as columns of matrix (denote by X which have the dimension $n \times p$):



X: Data of frames ($\in \mathbb{R}^{n \times p}$).
Robust PCA Formulation problem that we want to find L and S such:

$$X = L + S$$

L: Stationary background as low rank matrix (rank one).
S: The moving objects in the foreground.

Originale Frame Low_Rank " L " Sparse " S "

Consider the following convex minimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) + g(Ax)$$

Can be written us:

$$\min_{x,y} f(x) + g(y)$$
$$s.t. \quad Ax = y$$

The augmented Lagrangian:

$$\begin{aligned}\mathcal{L}_\rho(x, y, \lambda) &= f(x) + g(y) + \langle \lambda, Ax - y \rangle + \frac{\rho}{2}\|Ax - y\|^2 \\ &= f(x) + g(y) + \frac{\rho}{2}\|Ax - y + \frac{1}{\rho}\lambda\|^2 - \frac{1}{2\rho}\|\lambda\|^2\end{aligned}$$

Alternative direction method of multipliers (ADMM):

$$\begin{aligned}x_{k+1} &= arg\min_x \mathcal{L}_\rho(x, y_k, \lambda_k) \\ y_{k+1} &= arg\min_y \mathcal{L}_\rho(x_{k+1}, y, \lambda_k) \\ \lambda_{k+1} &= \lambda_k + \rho\,(Ax_{k+1} - y_{k+1})\end{aligned}$$

Convergence

- The residual $\|Ax_{k+1} - y_{k+1}\|$ converge to 0 (with rate $O(\frac{1}{k})$).
- The objective function $(f(x_{k+1}) + g(y_{k+1}))$ converge to $f^* + g^*$(with rate $O(\frac{1}{k})$)

The problem can be solved via convex programming:

$$\min_{L,S} \quad \|L\|_* + \lambda\|S\|_1$$
$$subject \quad to \quad X = L + S$$

$\|\cdot\|_*$ is the Nuclear norm (Sum of the singular values).

$$\mathcal{L}_\rho(x, y, \lambda) = \|L\|_* + \lambda\|S\|_1 + \frac{\rho}{2}\|L + S - X + \frac{1}{\rho}\lambda\|_F^2 - \frac{1}{2\rho}\|\lambda\|_F^2$$

the scalar product define in $\mathbb{R}^{n\times p}$ by: $\langle A, B\rangle_F = trace(A^T B)$.
Define the norm: $\|A\|_F = \sqrt{trace(A^T A)}$

Applying ADMM to Robust PCA:

$$L_{K+1} = arg \min_{L} \left\{ \|L\|_* + \frac{\rho}{2} \|L + S_k - X + \frac{1}{\rho}\Lambda_k\|_F^2 \right\}$$

$$S_{K+1} = arg \min_{S} \left\{ \lambda \|S\|_1 + \frac{\rho}{2} \|L_{k+1} + S - X + \frac{1}{\rho}\Lambda_k\|_F^2 \right\}$$

$$\Lambda_{k+1} = \Lambda_k + \rho(L_{K+1} + S_{K+1} - X)$$

Optimality conditions:

$$\nabla \left(\|L_{k+1}\|_* + \frac{\rho}{2} \|L_{k+1} + S_k - X + \frac{1}{\rho}\Lambda_k\|_F^2 \right) = 0$$

$A = U\Sigma V'$, Then $\nabla \|A\|_* = UV'$.
Indeed, we have:

$$\|A\|_* = trace(\Sigma)$$

Then,

$$\frac{\partial(\|A\|_*)}{\partial A} = \frac{trace(\partial\Sigma)}{\partial A}$$

$$\partial A = \partial(U)\Sigma V' + U\partial(\Sigma)V' + U\Sigma\partial(V')$$

implies that:

$$\partial\Sigma = U'\partial(A)V - U'\partial(U)\Sigma - \Sigma\partial(V)V'$$

then

$$trace(\partial\Sigma) = trace(U'\partial(A)V) - trace(U'\partial(U)\Sigma) - trace(\Sigma\partial(V)V')$$

since we have: $\partial(V'V) = 0$, Then $\partial V'V = -V'\partial V$, implies that:

$$trace(\Sigma \partial V V^{'}) = 0$$

(multiply a diagonal matrix with an anti-symmetric matrix is zero).
then, we get:

$$\frac{\partial(\|A\|_*)}{\partial A} = \frac{trace(U^{'}\partial(A)V)}{\partial A} = \frac{trace(VU^{'}\partial A)}{\partial A} = UV^{'}$$

If the svd of $L_{k+1} = U_{k+1}\Sigma_{k+1}V'_{k+1}$ then we have:

$$U_{k+1}V'_{k+1} + \rho\left(L_{k+1} + S_{k+1} - X + \frac{1}{\rho}\Lambda_k\right) = 0$$

$$U_{k+1}\left[\Sigma_{k+1} + \frac{1}{\rho}\mathbb{I}\right]V'_{k+1} = X - S_k - \frac{1}{\rho}\Lambda_k$$

if the svd of $X - S_k - \frac{1}{\rho}\Lambda_k = U_{k+1}\Upsilon V'_{k+1}$

which gives:

$$\Sigma_{k+1} + \frac{1}{\rho}\mathbb{I} = \Upsilon$$

finally:

$$L_{k+1} = U\left[\Upsilon - \frac{1}{\rho}\mathbb{I}\right]_{+} V'$$

Where $U\Upsilon V' = SVD(X - S_k - \frac{1}{\rho}\Lambda_k)$

$$
\begin{aligned}
L_{k+1} &= SVT_{\frac{1}{\rho}}\left(X - S_k - \frac{1}{\rho}\Lambda_k\right) \\
S_{k+1} &= ST_{\frac{\lambda}{\rho}}\left(X - L_{k+1} - \frac{1}{\rho}\Lambda_k\right) \\
\Lambda_{k+1} &= \Lambda_k + \rho(L_{k+1} + S_{k+1} - X)
\end{aligned}
$$

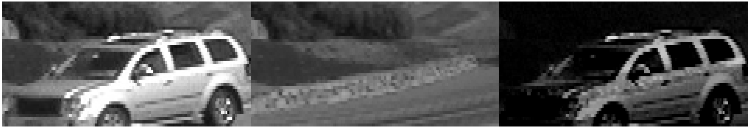
if for some matrix we have $A = U\Sigma V'$, then:

$$SVT_\lambda(A) = U[\Sigma - \lambda\mathbb{I}]_+ V'$$

and

$$
(ST_\lambda(X))_{i,j} = \begin{cases} X_{i,j} - \lambda & \text{if} \quad X_{i,j} > \lambda \\ 0 & \text{if} \quad |X_{i,j}| \leq \lambda \\ X_{i,j} + \lambda & \text{if} \quad X_{i,j} < -\lambda \end{cases}
$$

Simulation in Matlab for the video data: "visiontraffic.avi"
(Data base in Matlab)
- 40 frames.
- dimension of images (gray scale) 90×180
- We stop at the criterion $rank(L_{k+1}) = 1$.
frame number 30:

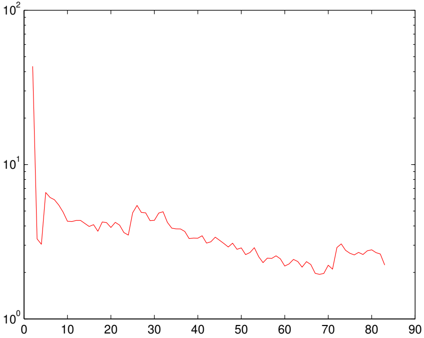


Original frame | Background (Low Rank) | Miving object (Sparse) frame number 20:

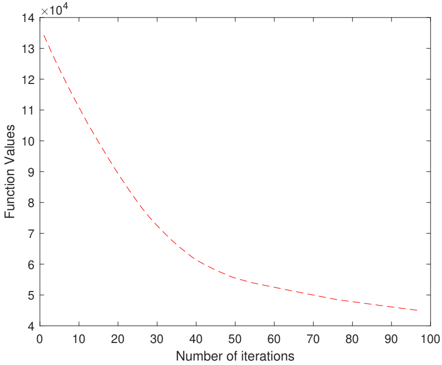
Remarks:

- converge when ρ is small, diverge if ρ is big.
- takes 25 minutes to show the result with laptop i3, 10^{th} Gen, 4GB ram.

$$\text{Residual} := norm(L_{new} + S_{new} - X)$$



Function value:$=\|L\|_* + \lambda\|S\|_1$



Simulation

matlab code:

```
clear all
R=VideoReader ('visiontraffic.avi'); %creates object v to read video data from the file
%%
X_Data=Matrix_video(R,161,200);%Put the 40 frames in a matrix
[n,m]=size(X_Data);
X = double(X_Data);
S=0*rand(n,m);
H=0*rand(n,m);
rho=0.005;
lambda=0.001;
L=rand(n,m);
epsilon=0.01;
i=0;
crit1=1; crit2=1;
risud=0;
rank_L=rank(L);
while( crit1 > epsilon  &&  crit2 > epsilon  &&  rank_L~=1)
 [Lnew,Snew,Hnew]=ADMM(X,S,H,rho,lambda);
 rank_L=rank(Lnew)
 crit1 =norm(Lnew+Snew-X)
 crit2 =rho*norm(Snew-S)
 risud=[risud crit1];
 % update rho
 if(crit1 >10*crit2) ,rho=2*rho;
 elseif(crit2 >10*crit1) , rho=rho/2;
 end
 i=i+1
```

Simulation

```
    L=Lnew;
    S=Snew;
    H=Hnew;
end
%% show the i eme image
for j =1:m
    Img_bg=Lnew(:,j);
    Img =Snew(:,j);
    Img_org=X_Data(:,j);
    figure(j)
    imshow([reshape(Img_org,60,90), uint8(reshape(Img_bg,60,90)), reshape(Img,60,90)]);
    hold on
end
figure 2
plot(risud(2:i),'-b');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ADMM function
function[L,S,K]=ADMM(X,S0,LM,rho,lambda)
% Compute the new Low rank
 [U,Sigm,V]=svd(X-S0-(1/rho)*LM);
 Sigm=max(Sigm-(1/rho),0);
 L=U*Sigm*V';
 %compute the sparse
 S=X-L-(1/rho)*LM;
 [n,m]=size(S);
 for i=1:n
     for j=1:m
```

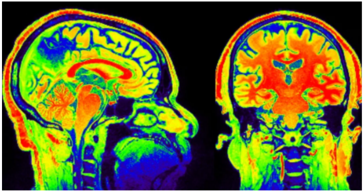
Simulation

```
    %compute the sparse
    S=X-L-(1/rho)*LM;
    [n,m]=size(S);
    for i=1:n
        for j=1:m
          if      (S(i,j)> lambda/rho)                    , S(i,j)=S(i,j) - (lambda/rho);
          elseif  (S(i,j)< -lambda/rho)                   ,S(i,j)=S(i,j) + (lambda/rho);%
          else                                            ,S(i,j)=0;
          end
        end
    end
    K=LM+rho*(L+S-X);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[M]=Matrix_video(vid,frame1,frame2)
Img = read(vid,frame1);
Img = rgb2gray(Img);
Img = Img(1:60,1:90);
M   = Img(:);
  for i=frame1+1:frame2
    frames = read(vid,i); % read picture in time i
    frames = rgb2gray(frames);
    frames = frames(1:60,1:90);
    M=[M frames(:)];
  end
end
```

Motivation

Sparse inverse covariance estimation: goal is to discover the direct connections between a set of nodes in a networked system based upon the observed node activities:

- Brain imaging data: how the brain works in health and disease by using the latest neuroimaging data (brain imaging).



- Let $X = (x_1, \cdots, x_n) \in \mathbb{R}^{n\times p}$ a multivariate normal distribution.
Σ: Correlation matrix associated to $X_{n\times p}$.

$$\Sigma = E\left[(X - E(X))^T (X - E(X))\right]$$

Σ^{-1}: determinate the conditional independence,
(since the density function:
$f(X) = Cst \times exp(X\Sigma^{-1}X) = \prod_i cst \times exp(x_i \Sigma_i^{-1} x_i)$ if Σ^{-1} is sparse).

- The complexity of computing the inverse of Σ is $o(n^3)$,
If n is big: Impossible of computing the inverse !

Approximate Σ^{-1} by solving the following minimization problem:

$$\min_{\Theta} -log \det(\Theta) + \langle \Theta, \Sigma \rangle + \lambda \|\Theta\|_1$$
$$s.t. \quad \Theta \succ 0$$

Optimality conditions:

$$-\Theta_*^{-1} + \Sigma + \lambda(sign(\Theta_*)) = 0$$

which gives that (if λ is chosen small enough):

$$\Sigma = \Theta_*^{-1}$$

ADMM for solving our graphical lasso problem:

$$\Theta_{k+1} = arg \min_{\Theta \succ 0} \left\{ -log \det(\Theta) + \frac{\rho}{2} \|\Theta - \Psi_k + \frac{1}{\rho}\Lambda_k + \frac{1}{\rho}\Sigma\|_F^2 \right\}$$

$$\Psi_{k+1} = arg \min_{\Psi} \left\{ \lambda \|\Psi\|_1 + \frac{\rho}{2} \|\Theta_{k+1} - \Psi + \frac{1}{\rho}\Lambda_k\|_F^2 \right\}$$

$$\Lambda_{k+1} = \Lambda_k + \rho \left(\Theta_{k+1} - \Psi_{k+1} \right)$$

Equivalent to:

$$\Theta_{k+1} = \mathcal{F}_\rho\left(\Psi_k - \frac{1}{\rho}\Lambda_k - \frac{1}{\rho}\sigma\right)$$

$$\Psi_{k+1} = ST_{\frac{\lambda}{\rho}}\left(\Theta_{k+1} + \frac{1}{\rho}\Lambda_k\right)$$

$$\Lambda_{k+1} = \Lambda_k + \rho\left(\Theta_{k+1} - \Psi_{k+1}\right)$$

For $X = U\Lambda V^{'} \succ 0$, once has:

$$\mathcal{F}_\rho(X) = \frac{1}{2}U\left[diag(\lambda_i + \sqrt{\lambda_i^2 + \frac{4}{\rho}})\right]V^{'}$$

1- S.Boyd, *Distributed Optimization and Statistical Learning via Alternating Direction Method of Multipliers*, 2010

2- J.Candes, and X. Li, *Robust Principal Component Analysis* , December 2009.

3- R.Mazumderand T.Hastie, *The Graphical lasso: New Insights and Alternatives*, 2008

4- H. Yin, X. Liu, *Gaussian Mixture Graphical Lasso with application to Edge Detection in Brain Networks*

5- J.Wright, *Robust principal component Analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, 2009

Thank you

for your attention