ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

DEEPTHI SUDHARSAN (CB.EN.U4AIE19022)

B.TECH CSE-AI, CEN DEPARTMENT, AMRITA SCHOOL OF ENGINEERING

ACKNOWLEDGEMENT

This project would not have been possible without the support and guidance of Dr. K P Soman, who has provided me with sufficient knowledge and guidelines even during these unprecedented and trying times.

I would also like to thank our Computer Science and Engineering (Artificial Intelligence) department for giving me this opportunity to nurture and hone my skills.

Furthermore, I would like to thank the Amrita Vishwa Vidyapeetham management for ample resources to avail my project needs and a platform for online lectures during this lockdown.

TABLE OF CONTENTS

ABSTRACT

Optimization is finding the best/optimum output by maximizing or minimizing a given function such that it also satisfies certain constraints. There are different types of optimization that are in use for a variety of purposes. **Convex and Non – convex optimizations** are two well-known types. The optimization is convex or non – convex based on whether or not its objective function or any of its constraints are convex or non-convex. Suppose, we are given the area of a lawn and we also know that it's perimeter is as small as possible and we have to find the dimensions of the lawn. This is an example of optimization.

The alternating direction method of multipliers (**ADMM**) is a powerful iterative algorithm that solves convex (Linear programming, Quadratic Programming etc) optimization problems by breaking them into smaller pieces. It is a modification of the **augmented Lagrangian** that partially updates the dual variables over iterations.

Augmented lagrangian is a popular method for solving constrained optimization problems. Augmented lagragian method that uses partial updates is none other than ADMM. The classic ADMM splits a complex problem into subproblems which is easy to solve.

ADMM on nonlinear equality problems are difficult to solve as the nonlinear equality constraint makes subproblems nonconvex. Until now, the conditions to the existence of optimal values of these subproblems remain a mystery. When it comes to ADMM on non-convex optimization problems, ways to find approximating methods for nonconvex functions and inequalities are still unknown as the solution may not be the global optimal value but guarantees 90 to 95 % accurate value near to the global optimal value.

*Keywords : Alternating Direction Method of Multipliers (**ADMM**), Optimization, Augmented Lagrangian*

ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

How to approach [convex] ADMM Problems? (Two Methods)

Method 1:

Given a convex ADMM problem where we have to
min *imize* f(x) + g(z)
s.t. Ax + Bz = c

Augmented Lagrangian is

$$L_{\rho}(x, z, y) = f(x) + g(z) + y^{T}(Ax + Bz - c) + \left(\frac{\rho}{2}\right)\|Ax + Bz - c\|_2^2$$

Let us look at what the terms of the augmented lagragian function are :

f(x) + g(z) is the objective function

y^T is the transpose of the lagrangian multiplier vector

$Ax + Bz - c$ is the constraint the objective function is subjected to

The above 3 terms constitute the general lagrangian function

$\frac{\rho}{2}$ is the penalty parameter

$\|Ax + Bz - c\|_2^2$ is the norm of the vector of equality constraints

It is the same as $(Ax + Bz - c)^T(Ax + Bz - c)$

In the ADMM iteration, firstly, we minimize in x

$x^{k+1} := \text{argmin}_x L_{\rho}(x, z^k, y^k)$ // k is the iteration parameter

Next, we minimize in z (we use the updated x from step 1)

$z^{k+1} := \text{argmin}_x L_\rho(x^{k+1}, z, y^k)$

Lastly we do the dual update (updation of multipliers),

$y^{k+1} = \text{y}^k + \rho(\text{Ax}^{k+1} + \text{Bz}^{k+1} - \text{c})$

ρ is the step size. Both step size and penalty parameter are assumed to be same to get simple expression with only one hyper parameter

$(\text{Ax}^{k+1} + \text{Bz}^{k+1} - \text{c})$ is gradient of L_ρ w.r.t. y

Let us consider the same minimization problem with different equality constraint

minimize f(x) + g(z)
s.t. x-z = 0

$$L_\rho(x, y, z) = f(x) + g(z) + y^T(x - z) + \left(\rho/2\right)\|x - z\|_2^2$$

It can also be written as,

$$L_\rho(x, y, z) = f(x) + g(z) + y^T x - y^T z + \left(\rho/2\right)\|x - z\|_2^2$$

$$x^{k+1} := \arg\min_x L_\rho\left(x, z^k, y^k\right)$$

$$z^{k+1} := \arg\min_z L_\rho\left(x^{k+1}, z, y^k\right)$$

$$y^{k+1} := y^k + \rho\left(x^{k+1} - z^{k+1}\right)$$

Note: *W*e are substituting the lagrangian equation form
into the ADMM iteration equations ; $\left\langle y^k, x\right\rangle = (\text{y}^k)^T x$; $\left\langle y^k, z\right\rangle = (\text{y}^k)^T z$

$$\Rightarrow x^{k+1} = \arg\min_x \left(f(x) + \left\langle y^k, x \right\rangle + \frac{\rho}{2} \left\| x - z^k \right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_z \left(g(z) - \left\langle y^k, z \right\rangle + \frac{\rho}{2} \left\| x^{k+1} - z \right\|_2^2 \right)$$

$$y^{k+1} := y^k + \rho \left(x^{k+1} - z^{k+1} \right)$$

We combine 2^{nd} *and* 3^{rd} *term in the first two optimization into a single term*

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{\rho}{2} \left\| x - z^k + \frac{1}{\rho} y^k \right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_z \left(g(z) + \frac{\rho}{2} \left\| x^{k+1} - z + \frac{1}{\rho} y^k \right\|_2^2 \right)$$

$$y^{k+1} = y^k + \rho \left(x^{k+1} - z^{k+1} \right)$$

Let us simplify the equations by substituting

$$u^k = \frac{1}{\rho} y^k, \quad \lambda = \frac{1}{\rho}$$

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{1}{2\lambda} \left\| x - (z^k - u^k) \right\|_2^2 \right)$$

$$z^{k+1} = \arg\min_z \left(g(z) + \frac{1}{2\lambda} \left\| z - (x^{k+1} + u^k) \right\|_2^2 \right)$$

$$y^{k+1} = y^k + \rho \left(x^{k+1} - z^{k+1} \right)$$

We get the proximal algorithm equations

$$\Rightarrow x^{k+1} := prox_{\lambda f} \left(z^k - u^k \right)$$

$$z^{k+1} := prox_{\lambda g} \left(x^{k+1} + u^k \right)$$

$$u^{k+1} := u^k + x^{k+1} - z^{k+1}$$

$$Note:$$

$$Proximal\ operator\ of\ a\ function = \underset{f}{prox}(v) = \arg\min_x \left(f(x) + \frac{1}{2}\|x - v\|_2^2 \right)$$

$$prox_{\lambda f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\lambda}\|x - v\|_2^2 \right) where\ \lambda > 0$$

Now that we have a simplified model, we can start with augmented Lagrangian functions of the form

$$L_\lambda(x, u, z) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

With these simplified equations , we can easily write first two optimization equations quite easily.

Method 2:

Another form of ADMM is where the equality constraints actually lie over a region. Let us look at an example,

$$\begin{aligned} &\min \quad f(x) \\ &subject\ \ to\ \ x \in C \end{aligned}$$

We convert it into

$$\begin{aligned} &\min \quad f(x) + g(z) \\ &subject\ \ to\, x - z = 0 \end{aligned}$$

Where,

g(z) is the indicator function and

$$g(z) = \begin{cases} 0 & if\ z \in C \\ \infty & otherwise \end{cases}$$

Augmented Lagrangian function can be written as

$$L_\lambda(x,u,z) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{1}{2\lambda}\|x - (z^k - u^k)\|_2^2 \right)$$

$z^{k+1} = \Pi_c(x^{k+1} + u^k)$ where, Π_c stands for projection onto convex set C

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Some Examples where ADMM comes into use

CONVEX OPTIMIZATION EXAMPLES :

(1) LASSO :

$$\min \ \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$$

$$subject\ \ to\, x - z = 0$$

$U \sin g$ ADMM form 1

$$L_\rho(x,y,z) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 + y^T(x - z) + \left(\rho/2\right)\|x - z\|_2^2$$

$$x^{k+1} = (A^T A - \rho I)^{-1}(A^T b - \rho(z^k - y^k))$$

$$z^{k+1} = S_{\frac{\lambda}{\rho}}\left(x^{k+1} - \frac{y^k}{\rho} \right)$$

where $S_{\frac{\lambda}{\rho}}$ is the soft thresholding operator which is

defined as the solution of a scalar optimization problem

$$y^{k+1} = y^k + \rho(x^{k+1} - z^{k+1})$$

Code:

Lasso function:

```
function x = lasso(A, b, lambda, rho)

MAX_ITER = 10;
RELTOL   = 1e-2; %error tolerance for ADMM

[m, n] = size(A);
Ata = A'*A;
Atb = A'*b;

x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

for k = 1:MAX_ITER

    % x-update
    x_1 = inv(Ata - rho*eye(n)) * (Atb - rho * (z -u));
    % z-update
    z_1 = (lambda/rho)*(x_1-(u/rho));
    % u-update
    u_1 = u + RELTOL*(x_1 - z_1);

    z = z_1;
    x = x_1;
    u = u_1;

end
end
```

Creating the problem and calling the function

```
m = 4;        % number of examples
n = 5;        % number of features
p = 0.05;      % p = 0.05 is the sparsity density
x0 = sprandn(n,1,p);
A = randn(m,n);
A = A*spdiags(1./sqrt(sum(A.^2))',0,n,n); % normalize
columns
b = A*x0 + sqrt(0.001)*randn(m,1);
lambda = 1;
x = lasso(A, b, lambda, 1.0)
```

<u>Sample Output</u>:

```
x =

   1.0e+08 *

   -5.2996

   -4.5646

    3.3924

    0.7834

    2.9229
```

NOTE: For small 'n' , we can use both " relaxation and shrinkage method of solving " and the above method without them for solving and getting the same answer. For large value of 'n', relaxation and shrinkage method of solving converges faster. Shrinkage will come if l1 norm there in the formulation. Relaxation is for faster convergence similar to the Cholesky decomposition for faster solution.

(2) Linear Programming (LP):

$$\min_x imize\ c^T x$$
$$subject\ \text{to}\ Ax = b,\ x \ge 0$$

Let us write it in the form

$$\min \quad f(x) + g(z)$$
$$subject\ \ to\, x - z = 0$$

where $\mathrm{f(x)} = \mathrm{c}^T x$

$$g(z) = \begin{Bmatrix} 0 & if\ z \in C \\ \infty & otherwise \end{Bmatrix}$$

$$L_\lambda(x,u,z) = f(x) + g(z) + \left(\frac{1}{2\lambda}\right)\|x - z + u\|_2^2$$

$$x^{k+1} = \underset{x:Ax=b}{\arg\min}\left(f(x) + \frac{1}{2\lambda}\|x - z^k + u^k)\|_2^2\right)$$

$$z^{k+1} = x^{k+1} + u^k$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Code:

<u>LP function:</u>

```
function z = linprog(c,A,b,rho,alpha)

MAX_ITER = 10;

%rho is the augmented Lagrangian parameter.
%sol is returned in vector z
% alpha is the over-relaxation parameter(typical values
for alpha are
% between 1.0 and 1.8).
[m n] = size(A);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);
for k =1:MAX_ITER
    %x-update
    tmp = [rho*eye(n), A';A, zeros(m)]\[rho*(z-u)-c;b];
    x = tmp(1:n);

    %z-update with relaxation
    zold = z;
    x_hat = alpha*x+(1 - alpha)*zold;
    z = pos(x_hat + u);

    u = u + (x_hat - z);
end
end
```

Creating the problem and calling the function:

```
n = 5;   % dimension of x
m = 4;   % number of equality constraints

c  = rand(n,1) + 0.5; % create nonnegative price vector
with mean 1
x0 = abs(randn(n,1)); % create random solution vector

A = abs(randn(m,n));  % create random, nonnegative
 matrix A
b = A*x0;

linprog(c, A, b, 1.0, 1.0)
```

Sample Output:

```
ans =

    0.5715

    1.1079

    1.4083

    0.6878

    0.6601
```

(3) Quadratic Programming :

$$\min imize\ \frac{1}{2}x^T Px + q^T x + r$$

subject to Ax = b, x $\geq$ 0

where,

P - m $\times$ n symmetric positive semidefinite matrix

$q \in R^n$; r $\in$ R

A - m $\times$ n matrix of rank m

Converting it into ADMM form

minimize f(x) +g(z)

subject to x - z = 0

where,

f(x) = $\frac{1}{2}x^T Px + q^T x + r$

and g(z) is the indicator function

ADMM Iteration updates:

$$x^{k+1} = \arg\min_x \left(f(x) + \left(\frac{\rho}{2}\right) \|x - z^k + u^k\|_2^2 \right)$$

$x - update$ involves solving the KKT equations

$$\begin{pmatrix} P+\rho I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x^{k+1} \\ y \end{pmatrix} = \begin{pmatrix} -q+\rho(z^k - u^k) \\ b \end{pmatrix}$$

$$z^{k+1} = (x^{k+1} + u^k)$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

Code:

<u>QP function:</u>

```
function x = qsolve1(P,q,A,b,rho,tolr,tols,iternum)
m=size(A,1);
n=size(P,1);
u=ones(n,1);
u(1,1)=0;
z=ones(n,1);
k=0;
nr=1; ns=1;

% convergence is controlled by norm nr of the primal
 residual r
% and the norm ns of the dual residual s

while(( k <= iternum) && (ns > tols || nr < tolr))
    z0=z;
    k=k+1;
    %KKT Matrix LHS
    KK = [P + rho*eye(n) A';A zeros(m,m)];
```

```
        %KKT Matrix RHS
        bb = [-q + rho*(z-u); b];
        %Solving KKT Matrix
        xx=KK\bb;
        %ADMM Updates
        x = xx(1:n);
         z = poslin(x+u); %Positive linear transfer
         function
        u = u + x - z;
        % To test stopping criterion
        r = x - z;  %primal residual
        nr = sqrt(r'*r); %norm of primal residual
        s = rho*(z-z0); % dual residual
        ns = sqrt(s'*s); %norm of dual residual
    end
    end
```

Creating the problem and calling the function:

```
    clc;
    clear all;
    close all;
    P2 = [4 1 0 0;1 4 1 0;0 1 4 1;0 0 1 4];
    q2 = [-4;-4;-4;-4];
    A2 = [1 1 -1 0;1 -1 -1 0];
    b2=[0;0];
    MAX_ITER = 100;
    rho2 = 10;
    tr = 10^(-12);
    ts = 10^(-12);

    Fvalue = qsolve1(P2,q2,A2,b2,rho2,tr,ts,MAX_ITER);
    xvalue = Fvalue(1)
    yvalue = Fvalue(2);
    zvalue = Fvalue(3);
     tvalue = Fvalue(4);
```

Sample Output:

```
    xvalue =

             0.9032
```

(4) Intersection of Polyhedral :

$$\min_{x,z} f(x) + g(z)$$

$$subject\ to\ x - z = 0$$

$$f(x) = \begin{cases} \infty \ if \ A_1 x > b_1 \\ 0 \ if \ A_1 x \le b_1 \end{cases}$$

$$g(z) = \begin{cases} \infty \ if \ A_2 z > b_2 \\ 0 \ if \ A_2 z \le b_2 \end{cases}$$

Augmented Lagrangian formulation
and ADMM updates

$$L(x, y, z) = f(x) + g(z) + y^T (x - z) + \frac{\rho}{2}\|x - z\|_2^2$$

$$x^{k+1} = \underset{x}{\text{argmin}}\ f(x) + y_k^T x + \frac{\rho}{2}\|x - z^k\|_2^2$$

$$z^{k+1} = \underset{z}{\text{argmin}}\ g(z) - y_k^T z + \frac{\rho}{2}\|x^{k+1} - z\|_2^2$$

$$\Rightarrow$$

$$x^{k+1} = \underset{x}{\text{argmin}}\ f(x) + \frac{\rho}{2}\left\|x - z^k + \frac{y^k}{\rho}\right\|_2^2$$

$$z^{k+1} = \underset{x}{\text{argmin}}\ g(z) + \frac{\rho}{2}\left\|x^{k+1} - z + \frac{y^k}{\rho}\right\|_2^2$$

$$Letting\ \frac{y^k}{\rho} = u^k \text{we get,}$$

$$x^{k+1} = \underset{x}{\text{argmin}}\ f(x) + \frac{\rho}{2}\|x - (z^k - u^k)\|_2^2$$

$$z^{k+1} = \underset{x}{\text{argmin}}\ f(x) + \frac{\rho}{2}\|z - (x^{k+1} + u^k)\|_2^2$$

The solution :

x^{k+1} is projection of vector $(z^k - u^k)$ on to $A_1 x \le b$

z^{k+1} is projection of vector $(x^{k+1} + u^k)$ on to $A_2 x \le b$

$u - update$

$u^{k+1} = u^k + (x^{k+1} - z^{k+1})$

Code:

Intersection of Polyhedral function:

```
function x = polyhedra_intersection(A1, b1, A2, b2, alpha)

MAX_ITER = 5;
n = size(A1,2);
x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

for k = 1:MAX_ITER

    % x-update
    % use cvx to find point in first polyhedra
    cvx_begin quiet
        variable x(n)
        minimize (sum_square(x - (z - u)))
        subject to
            A1*x <= b1
    cvx_end

    % z-update with relaxation
    zold = z;
    x_hat = alpha*x + (1 - alpha)*zold;
    % use cvx to find point in second polyhedra
    cvx_begin quiet
        variable z(n)
        minimize (sum_square(x_hat - (z - u)))
        subject to
            A2*z <= b2
    cvx_end

    u = u + (x_hat - z);
```

```
end

end
```

Creating the problem and calling the function:

```
n = 5;        % dimension of variable
m1 = 10;      % number of faces for polyhedra 1
m2 = 12;      % number of faces for polyhedra 2

c1 = 10*randn(n,1);           % center of polyhedra 1
c2 = -10*randn(n,1);          % center of polyhedra 2
% pick m1 n m2  random directions with different magnitudes
for A1 n A2
% resp

% the value of resp b is found by traveling from the center
along the normal
% vectors in resp A and taking its inner product with resp
A.

A1 = diag(1 + rand(m1,1))*randn(m1,n);
b1 = diag(A1*(c1*ones(1,m1) + A1'));

A2 = diag(1 + rand(m2,1))*randn(m2,n);
b2 = diag(A2*(c2*ones(1,m2) + A2'));

% find the distance between the two polyhedra--make sure
they overlap by
% checking if the distance is 0, if not expand A1 and A2 by
a little more than half the
% distance
cvx_begin quiet
    variables x(n) y(n)
    minimize sum_square(x - y)
    subject to
        A1*x <= b1
        A2*y <= b2
cvx_end

if norm(x-y) > 1e-4
    A1 = (1 + 0.5*norm(x-y))*A1;
    A2 = (1 + 0.5*norm(x-y))*A2;
```

```
    % recompute b's as appropriate
    b1 = diag(A1*(c1*ones(1,m1) + A1'));
    b2 = diag(A2*(c2*ones(1,m2) + A2'));
end

x = polyhedra_intersection(A1, b1, A2, b2, 1.0)
```

<u>Sample Output</u>:

```
x =

   1.0e-04 *

    0.1139

   -0.1486

    0.0945

   -0.0014

   -0.1195
```

(5) Total Variation Minimization :

$$\text{minimize } \frac{1}{2}\|x-b\|_2^2 + \lambda\sum_{i=1}^{n-1}|x_{i+1}-x_i|, \quad x \in R^n$$

$$\Rightarrow \text{minimize} \frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1$$

$$\textit{Subject} \text{ to } z = Dx$$

Augmented Lagrangian is

$$L(x,z,y) = \frac{1}{2}\|x-b\|_2^2 + \lambda\|z\|_1 + y^T(Dx-z) + \frac{\rho}{2}\|Dx-z\|_2^2$$

Update x

$$x - b + D^T y^k + \rho D^T \left(Dx - z^k \right) = 0$$

$$x^{k+1} = (I + \rho D^T D)^{-1}(b + \rho D^T (z^k - \frac{1}{\rho} y))$$

On replacing $\frac{1}{\rho} y^k = u^k$

$$x^{k+1} = (I + \rho D^T D)^{-1}(b + \rho D^T (z^k - u^k))$$

Update z:

$$L(x^{k+1}, z, y^k) = \lambda \|z\|_1 - \left(y^k \right)^T z + \frac{\rho}{2} \left\| Dx^{k+1} - z \right\|_2^2$$

$$L(x^{k+1}, z, y^k) = \|z\|_1 + \frac{\rho}{2\lambda} \left\| Dx^{k+1} - z + \frac{y^k}{\rho} \right\|_2^2$$

$$z^{k+1} = S_{\lambda/\rho} \left(Dx^{k+1} + \frac{y^k}{\rho} \right)$$

On replacing $\frac{1}{\rho} y^k = u^k$

$$z^{k+1} = S_{\lambda/\rho} \left(Dx^{k+1} + u^k \right)$$

Update u:

$$y^{k+1} = y^k + Dx^{k+1} - z^k$$

Code:

<u>Total Variation function:</u>

```
function x = total_variation(b, lambda, rho,alpha)
MAX_ITER = 1000;
n = length(b);
e = ones(n,1);
```

```
D = spdiags([e -e], 0:1, n,n);  %Sparse matrix formed from
diagonals

x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);

I = speye(n);
DtD = D'*D;

for k = 1:MAX_ITER

    % x-update
    x = (I + rho*DtD) \ (b + rho*D'*(z-u));

    % z-update with relaxation
    zold = z;
    Ax_hat = alpha*D*x +(1-alpha)*zold;
    a = Ax_hat + u;
    kappa = lambda/rho;
    z =  max(0, a-kappa) - max(0, -a-kappa);

    % u-update
    u = u + Ax_hat - z;
end
end
```

Creating the problem and calling the function:

```
% Total variation denoising with random data
clc;
clear all;
close all;

n = 2;

x0 = ones(n,1);
for j = 1:3
    idx = randsample(n,1);
    k = randsample(1:10,1);
    x0(ceil(idx/2):idx) = k*x0(ceil(idx/2):idx);
end
b = x0 + randn(n,1);
```

```
lambda = 5;

x = total_variation(b, lambda, 1.0,1.0)
```

Sample Output:

```
x =

   18.0913

    2.7901
```

NON - CONVEX OPTIMIZATION EXAMPLE:

(6) Regressor Selection (Non convex optimization) :

$$x^* = \underset{x}{\operatorname{argmin}} \ \frac{1}{2}\|Ax - b\|_2^2$$

subject to card(x) $\leq$ k , where card(x) is the number of non zero entities

Rewriting it in the form

$$\min_{x,z} \ f(x) + g(z) \ = \frac{1}{2}\|Ax - b\|_2^2 \ + \mathrm{g(z)}$$

subject to $x - z = 0$

$$g(z) = \begin{cases} 0 \text{ if card(z)} \leq \text{k} \\ \infty, \text{ if card(z)} > \text{k} \end{cases}$$

ADMM formulation :

Augmented Lagrangian is: $L(x,z,u) = f(x) + g(z) + \frac{1}{2\lambda}\|x - z + u\|_2^2$

$$\Rightarrow L(x,z,u) = \frac{1}{2}\|Ax - b\|_2^2 \ + g(z) + \frac{1}{2\lambda}\|x - z + u\|_2^2, \qquad \text{where } u=y/\sigma$$

Update x:

$$x^{k+1} = \arg\min_{x} \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2\lambda}\|x - z^k + u^k\|_2^2$$

Let $\rho = \frac{1}{\lambda}$

$A^T(Ax - b) + \rho(x - z^k + u^k) = 0$ vector

$(A^T A + \rho I)x = A^T b + \rho(z^k - u^k)$

$x^{k+1} = (A^T A + \rho I)^{-1}(A^T b + \rho(z^k - u^k))$

L(z)=$g(z) + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2$

From the term $\|x^{k+1} - z + u^k\|_2^2$, we get z and then project (select k largest and put all other to zero)to make indicator function g(z)=0

$z^{k+1} = keep\_l\arg est(x^{k+1} + u^k)$

Update u :

The original lagrangian multiplier term is u$^T(x - z)$

The gradient w.r.t. u is $x - z$

$\Rightarrow u^{k+1} = u^k + (x^{k+1} - z^{k+1})$

Code:

Regressor Selection function (LASSO):

```
function x = regressor_sel(A, b, K, rho)

[m, n] = size(A);
MAX_ITER = 10;

% save a matrix-vector multiply
Atb = A'*b;

x = zeros(n,1);
z = zeros(n,1);
u = zeros(n,1);
```

```
% cache the factorization
[L U] = factor(A, rho);


for k = 1:MAX_ITER

    % x-update
    q = Atb + rho*(z - u);     % temporary value
    if( m >= n )     % if skinny
       x = U \ (L \ q);
    else             % if fat
       x = q/rho - (A'*(U \ ( L \ (A*q) )))/rho^2;
    end

    % z-update with relaxation
    zold = z;
    z = keep_largest(x + u, K);

    % u-update
    u = u + (x - z);
end
end

function z = keep_largest(z, K)
    [val pos] = sort(abs(z), 'descend');
    z(pos(K+1:end)) = 0;
end

function [L U] = factor(A, rho)
    [m, n] = size(A);
    if ( m >= n )     % if skinny
       L = chol( A'*A + rho*speye(n), 'lower' );
    else             % if fat
       L = chol( speye(m) + 1/rho*(A*A'), 'lower' );
    end

    % force matlab to recognize the upper / lower
triangular structure
    L = sparse(L);
    U = sparse(L');
end
```

Creating the problem and calling the function:

```
m = 4;        % number of examples
n = 3;        % number of features
p = 100/n;       % sparsity density

% generate sparse solution vector
x = sprandn(n,1,p);

% generate random data matrix
A = randn(m,n);

% normalize columns of A
A = A*spdiags(1./sqrt(sum(A.^2))', 0, n, n);

% generate measurement b with noise
b = A*x + sqrt(0.001)*randn(m,1);

x = regressor_sel(A, b, p*n, 1.0)
```

Sample Output:

```
x =

    0.7423

    0.0089

    1.3611
```

Other Notable Applications of ADMM

1. Some applications of Distributed Reinforcement Learning
2. Filter Denoising
3. Image Restoration
4. Decentralized demand response method in electric vehicle virtual power plant.
5. Used for maximizing Weight Pruning
6. Energy Management of ancillary systems
7. ADMM based privacy-preserving decentralized optimization
8. Approach to informative trajectory planning for multi target tracking

REFERENCES

1. Unconstrained and Constrained Optimization Algorithms – Dr Soman K.P
2. https://www.youtube.com/watch?v=Xg0ozgCXXB8
3. https://www.youtube.com/watch?v=h8YKBVmRW-E
4. https://web.stanford.edu/~boyd/papers/pdf/admm_slides.pdf
5. https://www.youtube.com/watch?v=h8YKBVmRW-E
6. https://www.youtube.com/watch?v=1tyl_F8j3wA
7. https://scs.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=26607b7a-8a6a-49a6-99f1-a99801302a76
8. https://www.youtube.com/watch?v=05xOrD_-wvQ
9. https://web.stanford.edu/~boyd/papers/admm/
10. https://en.wikipedia.org/wiki/Augmented_Lagrangian_method#Alternating_direction_method_of_multipliers
11. https://arxiv.org/pdf/1705.03412.pdf
12. https://arxiv.org/pdf/1709.05747.pdf
13. https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
14. https://en.wikipedia.org/wiki/Augmented_Lagrangian_method#General_method
15. http://pwp.gatech.edu/ece-jrom/wp-content/uploads/sites/436/2017/07/16-admm.pdf
16. https://scholar.google.co.in/scholar?start=20&q=notable+applications+of+ADMM&hl=en&as_sdt=0,5&as_vis=1