TP 4 : MCP

Stéphane Canu

Octobre 2015, ASI, INSA Rouen

Le but du TP est d'étudier une autre méthode de sélection de variables, le *minimax concave penalties* (MCP), dans le cadre de la régression sur des données partiellement réelles. Pour le faire fonctionner, vous êtes supposé avoir déjà installé CVX (que vous pourrez télécharger à cette adresse : `http://cvxr.com/cvx/`)

Ex. 1 — le cout le gradient et les condition d'optimalité du MCP

1. Ecrire une fonction matlab `cout_mcp` permettant de calculer le cout MCP d'un problème de régression

$$J_{\text{MCP}}(\beta) \;=\; \tfrac{1}{2}\|y - X\beta\|^2 + \sum_{j=1}^{p} \mathbf{pen}_{\lambda,\gamma}(|\beta_j|)\;,$$

où $\mathbf{pen}_{\lambda,\gamma}$ est la pénalité du MCP définie par :

$$\mathbf{pen}_{\lambda,\gamma}(t) = \lambda \int_0^t \left(1 - \frac{x}{\gamma\lambda}\right)_+ dx = \begin{cases} \lambda t - \dfrac{t^2}{2\gamma} & \text{if } t \leq \gamma\lambda \\ \dfrac{\gamma\lambda^2}{2} & \text{else.} \end{cases}$$

```
function [cout ] = cout_mcp( Xi,yi,beta,lambda,gam )

cout = .5*(Xi*beta-yi)'*(Xi*beta-yi) + sum( (abs(beta)>gam*lambda)*gam*lambda^2/2 + (abs
    (beta)<=gam*lambda).*(lambda*abs(beta) - beta.^2/(2*gam)) );
end
```

2. Proposez une méthode permettant de tester votre calcul du cout
3. Ecrire une fonction matlab `grad_mcp` permettant de calculer le vecteur gradient du cout MCP d'un problème de régression

$$\partial_c J_{\text{MCP}}(\beta) \;=\; X^\top(X\beta - y) + \lambda \begin{cases} \alpha_j & \text{if } \beta_j = 0 \\ \text{sign}(\beta_j) - \dfrac{\beta_j}{\lambda\gamma} & \text{if } |\beta_j| \leq \lambda\gamma \\ 0 & \text{else}\;, \end{cases} \tag{1}$$

avec $\alpha_j \in [-1, 1]$.

```
function [g] = grad_mcp( X,y,beta,lambda,gam )

I0 = find(abs(beta)<sqrt(eps));
I2 = find(abs(beta)>lambda*gam);
I1 = 1:length(beta);
I1(sort([I0 ; I2])) = [];

Gls = X'*(X*beta-y);

g(I2) = Gls(I2);
g(I1) = Gls(I1) + lambda*sign(beta(I1)) - beta(I1)/gam;
g(I0) = (abs(Gls(I0) - beta(I0)/gam) - lambda).*(abs(Gls(I0) - beta(I0)/gam) > lambda);

end
```

4. Proposez une fonction permettant de tester votre calcul du gradient

Ex. 2 — le solveur component wise du MCP

1. La solution monovariable du MCP est trouvé à partir de la sous différentielle :

$$\frac{\partial J_{\text{MCP}}(\beta)}{\partial \beta_j} = X_{\bullet j}^\top (X\beta - y) + \begin{cases} \lambda \alpha_j & \text{if } |\beta_j| = 0 \\ sign(\beta_j) \max\left(0, \lambda - \frac{|\beta_j|}{\gamma}\right) & \text{else,} \end{cases}$$

avec $\alpha_j \in [-1, 1]$. Le vecteur $X_{\bullet j}$ denote la colonne j de la matrice X. Dans ce cas, $0 \in \frac{\partial J_{\text{MCP}}(\beta)}{\beta_j}$

$$\Leftrightarrow \quad \beta_j = \begin{cases} 0 & \text{if } |(X_{\bullet j}^\top X_{\bullet j})^{-1} X_{\bullet j}^\top \mathbf{r}| \leq \lambda \\ s_j \left(|(X_{\bullet j}^\top X_{\bullet j})^{-1} X_{\bullet j}^\top \mathbf{r}| - \lambda \right) \dfrac{\gamma}{\gamma - 1} & \text{if } |(X_{\bullet j}^\top X_{\bullet j})^{-1} X_{\bullet j}^\top \mathbf{r}| \leq \lambda\gamma \\ (X_{\bullet j}^\top X_{\bullet j})^{-1} X_{\bullet j}^\top \mathbf{r} & \text{sinon,} \end{cases}$$

avec s_j le signe de $(X_{\bullet j}^\top X_{\bullet j})^{-1} X_{\bullet j}^\top \mathbf{r}$ et $\mathbf{r} = X\beta - y - X_{\bullet j}\beta_j$ l'erreur résiduelle.

a) codez la en matlab

```
grad = - (X(:,ind(j))'*(X*beta-y - X(:,ind(j))*beta(ind(j))));
beta(ind(j)) = sign(grad)*max(0,min((abs(grad) - lambda)/(1-1/gam),abs(grad)));
```

b) Intégrez ce calcul dans une boucle permettant de parcourir toutes les variables dans un robre aléatoire

```
ind = randperm(p);
b0 = beta;
for j = 1:p
   grad = - (X(:,ind(j))'*(X*beta-y - X(:,ind(j))*beta(ind(j))));
   beta(ind(j)) = sign(grad)*max(0,min((abs(grad) - lambda)/(1-1/gam),abs(grad)));
end
```

c) intégrez ce code dans une boucle donc vous discuterez les conditions d'arrêt.

d) Ecrire une fonction matlab `MCP_CW` permettant de résoudre le MCP à l'aide d'une méthode de type composante par composante.

```
function [ beta ] = MCP_CW( X,y, lambda, gam, beta_init )
% function [ beta ] = MCP_CW( X,y, lambda, gam, beta_init )

nb_ite_max = 250;
tol = 10^-6;

beta = beta_init;
b0 = beta+1;

nb_ite = 0;
[n,p] = size(X);

while ( (max(abs(beta-b0)) > tol) & (nb_ite < nb_ite_max) )

   ind = randperm(p);
   b0 = beta;
   for j = 1:p
     grad = - (X(:,ind(j))'*(X*beta-y - X(:,ind(j))*beta(ind(j))));
     beta(ind(j)) = sign(grad)*max(0,min((abs(grad) - lambda)/(1-1/gam),abs(grad)));
   end
   nb_ite = nb_ite +1 ;

end
end
```

e) Ecrire une fonction permettent de tester votre fonction `MCP_CW`. Vérifiez que le cout diminue et que la solution vérifie bien les conditions d'optimalisés

Ex. 3 — le solveur DC du MCP

1. Based on this decomposition, the DC algorithm amounts to iteratively building a sequence by minimizing, for a given β^{old}, the following convex surrogate cost function

$$J_{\mathrm{DC}}(\beta) = \tfrac{1}{2}\|y - X\beta\|^2 + \lambda\|\beta\|_1 - \lambda \sum_{j=1}^{p} h'_{\lambda,\gamma}(|\beta_j^{old}|)\,|\beta_j|, \tag{2}$$

h' being the derivative of the Huber loss function. It turns out that in that case minimizing the DC cost function (2) can be seen as minimizing an adaptive Lasso.

$$\min_{\beta \in \mathbf{R}^p} \ \tfrac{1}{2}\|y - X\beta\|^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|, \tag{3}$$

with weights

$$w_j = \begin{cases} 0 & \text{if } |\beta_j^{old}| > \lambda\gamma \\ 1 - \dfrac{|\beta_j^{old}|}{\gamma\lambda} & \text{else} \end{cases}.$$

a) Ecrire une fonction matlab `MCP_DC` permettant de résoudre le MCP à l'aide d'une méthode de type DC.

```
function [ beta ] = MCP_DC( X,y, lambda, gam, beta_init )
%function [ beta ] = MCP_DC( X,y, lambda, gam, beta_init )
%
% This function solves the MCP regression problem using a DC approach
% This is also the MM, LLa and CCCP solution
%
% min .5||X beta - y||^2 + lambda S_j=1^p pen(|beta_j|)
% beta
%

nb_ite_max = 250;
tol = 10^-6;

beta = beta_init;
b0 = beta+1;

nb_ite = 0;
[n,p] = size(X);

H = [X'*X -X'*X;-X'*X X'*X];

while ( (max(abs(beta-b0)) > tol) & (nb_ite < nb_ite_max) )

    b0 = beta;

    w = max(0,1 - abs(beta)/gam/lambda); % computing the weights

    c = [X'*y - lambda*w; -X'*y - lambda*w];
    [xnew, pos] = monqpL(H,c,tol^2,0);
    Bpm = zeros(2*p,1);
    Bpm(pos) = xnew;
    beta = Bpm(1:p)-Bpm(p+1:end);

    nb_ite = nb_ite +1 ;
end
```

b) Ecrire une fonction permettent de tester votre fonction `MCP_DC`. Vérifiez que le cout diminue et que la solution vérifie bien les conditions d'optimalisés

Ex. 4 — Comparaison des solveur CW et DC du MCP

1. comparez les solutions des deux soldeurs sur le problème suivant :

```
n = 30;
p = 50;

Xi = randn(n,p);

r = .5; %1/1.25; % model 1 by zhou
 for (a = 1:p)
      for (b = 1:p)
        C(a,b) = r^abs(a-b);
      end
 end

Xi = Xi*chol(C); % la correlation qui tue

betaVrai = zeros(p,1);
betaVrai(1:10) = [1 2 3 4 5 -1 -2 -3 -4 -5];
sig = 0.5;

Xi = (Xi - ones(n,1)*mean(Xi))./(ones(n,1)*std(Xi,1));
Xi = Xi/sqrt(n);

yi = Xi*betaVrai + sig*randn(n,1);

lambda = .5;
gam = 2;
```

2. comparez aussi vos résultats à ceux de GIST[1]

[1] http://www.public.asu.edu/~jye02/Software/GIST/