TP 1 : Le lasso

Stéphane Canu

Septembre 2017, ASI, INSA Rouen

Le but du TP est d'étudier une méthode de sélection de variables, le Lasso[1], 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/`)

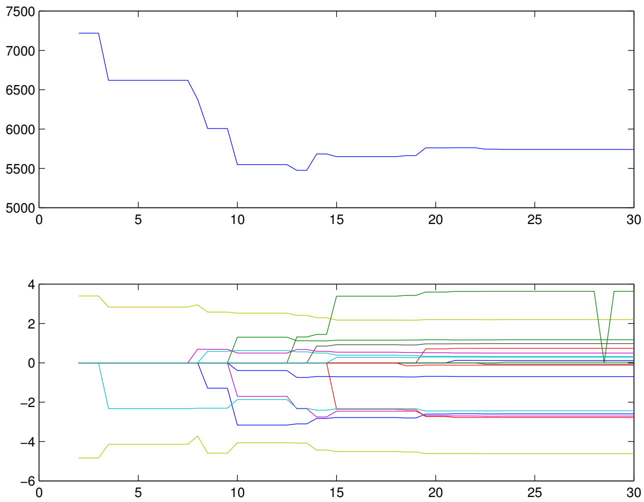

FIGURE 1 – Résultat du TP1

Ex. 1 — Le Lasso comme une méthode de sélection de variables

1. Génération des données du problème.
   a) Générez les données du problème en utilisant les données « Boston houssing »[2], auxquelles vous ajouterez $q = 5$ variables aléatoires (donc inutiles).

```
prepare_housing % load data
randn('seed',10);
[n,p] = size(X);
q = 5;          % adding useless variables
X = [X randn(n,q)];
p = p+q;
```

   b) Séparez les données disponible en deux sous ensembles d'apprentissage et de test de taille égale.

```
ind = randperm(n);
na = n/2;

Xi = X(ind(1:na),:);
yi = y(ind(1:na));

Xt = X(ind(na+1:end),:);
yt = y(ind(na+1:end));
```

   c) Calculez l'erreur de test de la méthode des moindres carrés

```
beta_mc = Xi\yi;
Erreur = (yt - Xt*beta_mc)'*(yt - Xt*beta_mc)
```

[1] `http://statweb.stanford.edu/~tibs/lasso.html`
[2] `https://archive.ics.uci.edu/ml/datasets/Housing`

2. Différentes manières de résoudre le problème du Lasso

   a) Ecrire une programme CVX résolvant, pour $k = 10$ et qui permet de connaitre la valeur du multiplicateur de Lagrange *lambda* à l'optimum.

$$\begin{cases} \min_{\beta \in \mathbb{R}^p} & \frac{1}{2}\|X\beta - y\|^2 \\ \text{avec} & \sum_{j=1}^{p} |\beta_j| \leq k \end{cases} \tag{1}$$

```
[na,p] = size(Xi);
k = 10;
cvx_begin
  cvx_precision best
  variables beta1(p)
  dual variable d
  minimize( norm(yi - Xi*beta1,2) )
  subject to
    d : sum(abs(beta1)) <= k;
cvx_end
```

   b) Ecrire une programme CVX résolvant la formulation suivant de Lasso, avec comme valeur de λ le multiplicateur de Lagrange du problème précédent à l'optimum.

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|X\beta - y\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

```
cvx_begin
  variables beta2(p)
  minimize( norm(yi - Xi*beta2,2) + d * sum(abs(beta2)) )
cvx_end
```

   c) résoudre le problème du Lasso (1) en réécrivant le cout comme une fonctionnelle quadratique de la forme

$$\frac{1}{2}\beta^\top D\beta + \beta^\top e$$

   où la matrice D et le vecteur e sont à préciser

```
D = Xi'*Xi;
ep = -yi'*Xi;
cvx_begin
  variables beta2(p)
  dual variable d2
  minimize( .5*beta2'*D*beta2 + ep*beta2 )
  subject to
    d2 : sum(abs(beta2)) <= k;
cvx_end
```

   d) réécrire le Lasso comme un programme quadratique sous sa forme standard.

$$\begin{cases} \min_{x \in \mathbb{R}^n} & \frac{1}{2}x^\top Hx + x^\top c \\ \text{avec} & Ax \leq b \end{cases} \tag{2}$$

```
  H = [Xi'*Xi -Xi'*Xi;-Xi'*Xi Xi'*Xi];
  c = [Xi'*yi ; -Xi'*yi];
  A = ones(2*p,1);
  b = k;
  l = 10^-12;
  verbose = 0;
```

e) proposez un code CVX permettant de résoudre le Lasso réécrit comme un QP standard.

```
cvx_begin
  variables Bpm(2*p)
  dual variable dpm
  minimize( .5*Bpm'*H*Bpm - c'*Bpm )
  subject to
      dpm : sum(Bpm) <= b;
      0 <= Bpm;
cvx_end
```

f) résoudre le Lasso réécrit comme un QP standard en utilisant CPLEX.

```
    xcplex = cplexqp(H,-c,[],[],A',k,0*c);
```

g) vérifiez que toutes les méthodes donnent le même résultat et comparez les temps de calcul.

```
betam = xcplex(1:p)-xcplex(p+1:2*p);
[beta1 beta2 Bpm(1:p)-Bpm(p+1:end) betam]
```

h) quelle est la plus rapide des méthodes ? A quoi sert CVX ?

3. Nous allons comparez les méthodes en terme d'erreur de test
   a) calculez les couts de la solution des moindres carrées et de la solution du lasso

```
    beta_mc = Xi\yi;
    err_mc = (yt - Xt*beta_mc)'*(yt - Xt*beta_mc);
    err_L = (yt - Xt*beta1)'*(yt - Xt*beta1);
```

   b) calculez les couts de la solution des moindres carrées calculée sur les variables sélectionnées par le Lasso.

```
    pos = find(abs(beta1)>0.000001);
    beta_mc = Xi(:,pos)\yi;
    err_Lmc = (yt - Xt(:,pos)*beta_mc)'*(yt - Xt(:,pos)*beta_mc);
```

   c) Comparez les couts des différents cout et commentez. Que peut on dire des gradients ?

```
    [err_mc err_L err_Lmc]
```

4. A la recherche du meilleur paramètre de régularisation λ
   a) écrire une boucle permettant de tester différentes valeurs de $\lambda \in [2, 30]$ par pas de $\frac{1}{2}$. On utilisera l'estimateur des moindres carrés sur les variables sélectionnées par le Lasso. On stockera les valeurs estimées de β dans une matrice B.

```
    K = [2:0.5:30];
    B = [];
    for i=1:length(K)
    k = K(i);
    [xnew, lambda, pos] = monqp(H,c,A,k,inf,l,verbose);
    ind = find(pos>p);
    sign = ones(length(pos),1);
    pos(ind) = pos(ind) - p;
    sign(ind) = -1;
    beta_mc = Xi(:,pos)\yi;
    err(i) = (yt - Xt(:,pos)*beta_mc)'*(yt - Xt(:,pos)*beta_mc);
    beta = 0*beta1;
    beta(pos) = beta_mc;
    B = [B beta];
    end;
```

   b) retrouver la meilleure estimation de λ (au sens de l'erreur de test)

```
    [v ind] = min(err);
    beta = B(:,ind);
    ind = find(abs(beta) < 0.000001);
```

```
Xi(:,ind) = [];
Xt(:,ind) = [];
beta_mc = Xi\yi;
Erreur = (yt - Xt*beta_mc)'*(yt - Xt*beta_mc)
```

c) affichez dans un graphique les résultats : l'erreur de test en fonction de λ en haut, et les valeures des coefficients de β toujours en fonction de λ en bas.

```
close all
figure(1)
subplot(2,1,1)
plot(K,err)
subplot(2,1,2)
plot(K,B')
```

5. **A vous de jouer :**
   a) Proposez un code Gurobi permettant de résoudre le Lasso réécrit comme un QP standard.
   b) Ecrire une fonction matlab $\beta \leftarrow$ `lasso`(X, y) efficace et qui sélectionne automatiquement un bon λ.