TP 2 : Le lasso adaptatif et la formulation duale

Stéphane Canu

Septembre 2015, ASI, INSA Rouen

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

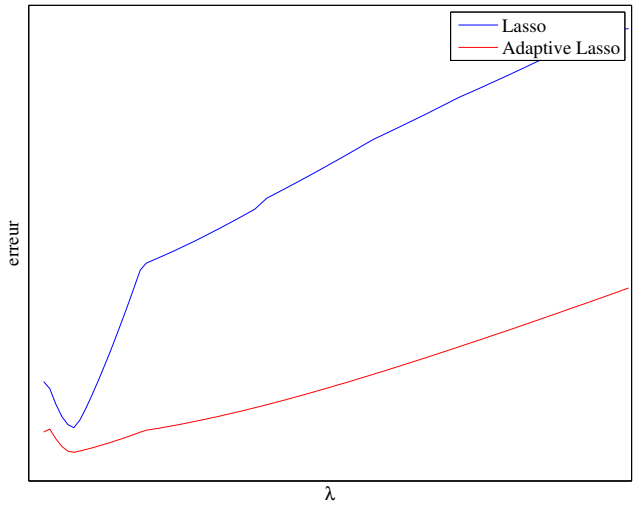

FIGURE 1 – Résultat du TP2

Ex. 1 — Dualité du Lasso et des variantes du Lasso

1. Génération des données du problème.
   a) Générez les données du problème $n = 50$ et p= 18.

```
clear
rand('seed',3);
randn('seed',3);

n = 50;
p = 18;

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

betaVrai = zeros(p,1);
betaVrai(1:10) = [1 2 3 4 5 1 2 3 4 5];
sig = 0.25;
yi = Xi*betaVrai + sig*randn(n,1);

[n,p] = size(Xi);
```

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

```
b_mc = (Xi'*Xi)\(Xi'*yi);
```

[1] `http://www.stat.wisc.edu/~shao/stat992/zou2006.pdf`

2. Le Lasso et son dual : vérifiez que les deux méthodes donnent le même résultat.
   a) Résoudre le Lasso comme nous l'avons fait lors du TP1

```
lambda = 2;
tic
cvx_begin
    variables betaL(p)
    minimize( .5* (yi - Xi*beta_lasso)'*(yi - Xi*betaL) + lambda * sum(abs(betaL)) )
cvx_end
```

   b) Calculez l'erreur du modèle

```
Err = norm(betaVrai-betaL);
fprintf('Lasso Primal: %10.2f - ', Err)
toc
```

   c) résoudre le dual du lasso à l'aide de CVX (version 1)

```
tic
cvx_begin
    variables alpha_cvx(n)
    dual variable d1
    minimize( .5*alpha_cvx'*alpha_cvx - alpha_cvx'*yi )
    subject to
        d1 : abs(Xi'*alpha_cvx) <= lambda;
cvx_end
beta_alpha = (Xi'*Xi)\Xi'*(yi-alpha_cvx);
Err = norm(betaVrai-beta_alpha);
fprintf('Lasso Dual: %10.2f - ', Err)
toc
```

   d) résoudre le dual du lasso à l'aide de CVX (version 2)

```
tic
cvx_begin
    variables beta_dual(p)
    dual variable d
    minimize( norm(Xi*beta_dual,2) )
    subject to
        d : abs(Xi'*(Xi*beta_dual - yi)) <= lambda;
cvx_end
Err = norm(betaVrai-beta_dual);
fprintf('Lasso Dual: %10.2f - ', Err)
toc
```

   e) résoudre le dual du lasso à l'aide de GUROBI (version 2)

```
clear model;
model.Q = .5*sparse(Xi'*Xi);
model.obj = zeros(p,1);
model.A = [sparse(Xi'*Xi);sparse(Xi'*Xi)];
model.sense = [repmat('<',p,1);repmat('>',p,1)];
clear params;
params.Presolve = 2;
params.TimeLimit = 100;
params.OutputFlag = 0;

model.rhs = [lambda + Xi'*yi;-lambda + Xi'*yi];
result = gurobi(model, params);
beta_gu = result.x;
```

   f)

```
fprintf(' Truth Least Square Lasso \n')
for i=1:length(beta_lasso)
    fprintf('%10.2f %10.2f %10.2f %10.2f %10.2f %10.2f \n',[betaVrai(i) b_mc(i)
        betaL(i) beta_alpha(i) beta_dual(i) beta_gu(i)]);
end
```

3. Elastic Net
   a) Calcul dans le primal à l'aide de CVX

```
mu = 1;
tic
cvx_begin
    variables beta_en(p)
    minimize( .5* (yi - Xi*beta_en)'*(yi - Xi*beta_en) + lambda * sum(abs(beta_en))
        + mu/2*norm(beta_en))
cvx_end
Err = norm(betaVrai-beta_en);
fprintf('El Net Primal: %10.2f - ', Err)
toc
```

   b) En utilisant votre une fonction matlab $\beta \leftarrow \texttt{lasso}(X, y)$

```
tic
beta_enL = monLasso([Xi ; sqrt(mu)*eye(p)],[yi;zeros(p,1)]);
Err = norm(betaVrai-beta_enL);
fprintf('El Net Primal: %10.2f - ', Err)
toc
```

4. Le lasso adaptatif
   a) Calculez les poids

```
w = 1./abs(b_mc); % set the weight
```

   b)

```
tic
cvx_begin
    variables beta_adapt_lasso(p)
    minimize( .5* (yi - Xi*beta_adapt_lasso)'*(yi - Xi*beta_adapt_lasso) + lambda *
        w'*abs(beta_adapt_lasso) )
cvx_end
Err = norm(betaVrai-beta_adapt_lasso);
fprintf('Adaptive Primal: %8.2f - ', Err)
toc
```

   c) en décomposant $\beta = \beta^+ - \beta^-$

```
tic
cvx_begin
    variables beta_cvx(p) beta_cvxm(p)
    dual variable d
    minimize( norm(yi - Xi*(beta_cvx-beta_cvxm),2) )
    subject to
        d : w'*(beta_cvx+beta_cvxm) <= b;
        beta_cvx >= 0;
        beta_cvxm >= 0;
cvx_end
fprintf('CVX Primal : ')
toc
beta_cvx = beta_cvx-beta_cvxm;
```

   d) en utilisant `monqp`

```
tic
H = [Xi'*Xi -Xi'*Xi;-Xi'*Xi Xi'*Xi];
c = [Xi'*yi ; -Xi'*yi];
l = 10^-9;
A = [w ; w];
b = w'*abs(beta_adapt_lasso);
[xnew, dual_var, pos] = monqp(H,c,A,b,inf,l,0);
fprintf('Adaptive monQP: ')
toc
b_mqp = zeros(2*p,1);
b_mqp(pos) = xnew;
b_mqp = b_mqp(1:p) - b_mqp(p+1:end);
```

e) En utilisant votre une fonction matlab $\beta \leftarrow \texttt{lasso}(X, y)$

```
tic
beta_adL = monLasso(Xi*diag(sqrt(w)),yi);
Err = norm(betaVrai-beta_adL);
fprintf('Adaptive monLasso: %6.2f - ', Err)
toc
```

f) dans le dual (1)

```
tic
cvx_begin
    cvx_precision best
    variables alpha_cvx(n)
    dual variable d
    minimize( .5*alpha_cvx'*alpha_cvx - alpha_cvx'*yi )
    subject to
        d : abs(Xi'*alpha_cvx) <= lambda*w;
cvx_end
fprintf('CVX Dual: ')
toc
beta_alpha = (Xi'*Xi)\Xi'*(yi-alpha_cvx);
```

g) dans le dual (2)

```
tic
cvx_begin
    cvx_precision best
    variables beta_dual(p)
    dual variable d
    minimize( norm(Xi*beta_dual,2) )
    subject to
        d : abs(Xi'*(Xi*beta_dual - yi)) <= lambda*w;
cvx_end
fprintf('CVX Dual 2: ')
toc
```

h) en utilisant Gurobi

```
clear model;
model.Q = sparse(Xi'*Xi);
model.obj = zeros(p,1);
model.A = [sparse(Xi'*Xi);sparse(Xi'*Xi)];
model.sense = [repmat('<',p,1);repmat('>',p,1)];
model.rhs = [lambda*w + Xi'*yi;-lambda*w + Xi'*yi];

clear params;
params.Presolve = 2;
params.TimeLimit = 100;
params.OutputFlag = 0;

result = gurobi(model, params);

beta_gu = result.x;
```

i) Affichez les résultats et vérifiez que toutes les méthodes donnent le même résultat.

```
fprintf(' Truth Least Square Lasso Adaptive Lasso \n')
for i=1:length(beta_lasso)
    fprintf('%10.2f %10.2f %10.2f %10.2f %10.2f %10.2f %10.2f %10.2f %10.2f \n',[
        betaVrai(i) b_mc(i) beta_lasso(i) beta_adapt_lasso(i) b_mqp(i) beta_cvx(i)
        beta_alpha(i) beta_dual(i) beta_gu(i)]);
end
```

5. Pour faire le dessin illustrant le TP2, il vous faut installez gurobi[2]

[2] http://www.gurobi.com/academia/for-universities

a)
```
Err = [];
Lambda = 0.025:0.01:1;
for i=1:length(Lambda)
    lambda = Lambda(i);
model.rhs = [lambda + Xi'*yi;-lambda + Xi'*yi];
    result = gurobi(model, params);
    beta_guT = result.x;
    Err = [Err norm(betaVrai-beta_guT)];
end
figure(2)
plot(Lambda,Err);
```

b)
```
Err = [];
for i=1:length(Lambda)
    lambda = Lambda(i);
    model.rhs = [lambda*w + Xi'*yi;-lambda*w + Xi'*yi];
    result = gurobi(model, params);
    beta_guT = result.x;
    Err = [Err norm(betaVrai-beta_guT)];
end
hold on
plot(Lambda,Err,'r');
hold off
legend('Lasso','Adaptive Lasso')
xlabel('\lambda')
ylabel('erreur')
```

6. Concrètement, entre le Lasso, l'*elastic net*, le Lasso adaptatif et le sélecteur de Dansig, la quelle de ces méthodes préconiseriez vous et dans quelles conditions ?