TP6: Algorithmes de descente de gradient proximale

Alain Rakotomamonjy

November 1, 2015

Description du TP

This practical session aims at writing generic first-order gradient algorithms that solve the sparse problem that occurs in variable selection in SVM, logistic regression or linear regression problems.

Ex. 1 — Some proximal operators

1. Proximal operators
   a) implement a function that computes the ℓ_1 operator `Proxl1`

```
function wprox=proxl1(w,lambda)

wprox=sign(w).*max(abs(w)-lambda,0)
```

   b) implement function that computes the group-lasso proximal operator (optional)

```
function wprox=proxgrouplasso(w,lambda,group)

% group is a vector of same size of w stating to which group
% w_i belongs

nbgroup=max(group); % count the number of group
wprox=w(size(w));
for i=1:nbgroup
    ind=find(group==i); % find all the variables in a given group
    wprox(ind)= max(1-lambda/norm(w(ind)),0)*w(ind);
end;
```

Ex. 2 — Proximal gradient algorithms

1. Proximal gradient algorithm for the Lasso
   a) Implement a proximal gradient algorithm for the Lasso problem that we already used in previous labs.

```
lambda=0.1;
w=zeros(d,1);
stepsize = 1/norm(X'*X); % choose L as a step, L being the norm of the Hessian
for i=1:5000
    grad= -X'*(y-X*w);
    w=w - stepsize*grad;
    w=proxl1(w,stepsize*lambda);
end;
```

   b) Check if the problem has been properly solved by verifying the optimality conditions

$$\begin{aligned} sign(w_j^\star) \neq 0 &\implies -\mathbf{x}_j^\top(\mathbf{y} - \mathbf{X}\mathbf{w}^\star) + \lambda sign(w_j) = 0 \\ sign(w_j^\star) = 0 &\implies |\mathbf{x}_j^\top(\mathbf{y} - \mathbf{X}\mathbf{w}^\star)| \leq \lambda \end{aligned}$$

```
epsi=1e-6;
indzero=find(abs(w)<epsi);
indnonzero=find(abs(w)>=epsi);
grad=-X'*(y-X*w);
exactOnZeros= max(abs(grad(indzero))-lambda);
exactOnNonZeros= max( abs(abs(grad(indnonzero)) - lambda));
```

c) Write a function `ProximalSparseRegression` that implements a proximal descent algorithm that stops when a stopping criterion based on the optimality conditions is reached.

```
function w=ProximalSparseRegression(X,y,lambda,epsi)
```

d) which algorithm is faster on your problem the coordinate descent algorithm or the proximal descent ones? what if we change λ and the number of active variables in the true model.

```
lambda=0.1;
epsi=1e-3;
tic
wprox=ProximalSparseRegression(X,y,lambda,epsi);
timingprox=toc
tic
xcd=CDsparseRegression(X,y,lambda,epsi);
timingcoordinate=toc
```

2. Sparse Support Vector Machines in the primal. Now we want to design a SVM that selects automatically the relevant variables for the decision function. For having a differentiable loss function, we use a squared Hinge loss.

$$\min_{\mathbf{w}} \frac{1}{2}(1 - Y(X\mathbf{w} + w_0))_+^\top (1 - Y(X\mathbf{w} + \mathbf{w}_0))_+ + \lambda \|\mathbf{w}\|_1$$

a) implement an iterative proximal gradient algorithm that solves this sparse squared Hinge loss problem. Apply it on the Housing dataset available on Moodle.

```
clear all
close all
% loading the data X y (training) xtest, ytest (test)
load('housing.mat');
% choose the penalty value. the larger lambda, the sparser the solution
lambda=1;
% compute stepsize $L$ the norm of the Hessian
augmentedMatrix=[X ones(size(X,1),1)];
stepsize=1/norm(augmentedMatrix'*augmentedMatrix);
w=zeros(size(X,2),1);
w0=0;
% proximal descent algorithm
Y=diag(y);
for i=1:5000
    % computing the gradient wrt w and w0
    loss=max(1-Y*(X*w + w0),0);
    gradw=-(Y*X)'*loss;
    gradw0=-(Y*ones(size(X,1),1))'*loss;
    % proximal step
    w=proxl1(w-stepsize*gradw,lambda*stepsize);
    w0=w0-stepsize*gradw0;
end;
% check how good your algorithm is doing on the test set
mean(sign(xtest*w+w0)==ytest)
```

b) Write a function `ProximalSparseSVM` that implements a proximal descent algorithm that stops when a stopping criterion based on the optimality conditions on $\mathbf{w}$ is reached according to a tolerance defined by `epsi`.

```
function [w,w0]=ProximalSparseSVM(X,y,lambda,epsi)
```

c) play with λ to figure out the best value that maximizes the test performance. How many non-zeros elements in $\mathbf{w}$ (out of the 113) do we have?

```
load('housing.mat');
lambda=0.5; % you have to change the lambda
epsi=0.01;
[w,w0]=ProximalSparseSVM(X,y,lambda,epsi);
mean(sign(xtest*w+w0)==ytest)
```

d) what would change in your function if you want to use group lasso, positive ℓ_1 penalty or positive weight SVM?