Advanced Topics in Numerical Analysis: Convex and Nonsmooth Optimization Problem Set 8

Abhijit Chowdhary
ac6361@nyu.edu

New York University — April 7, 2020

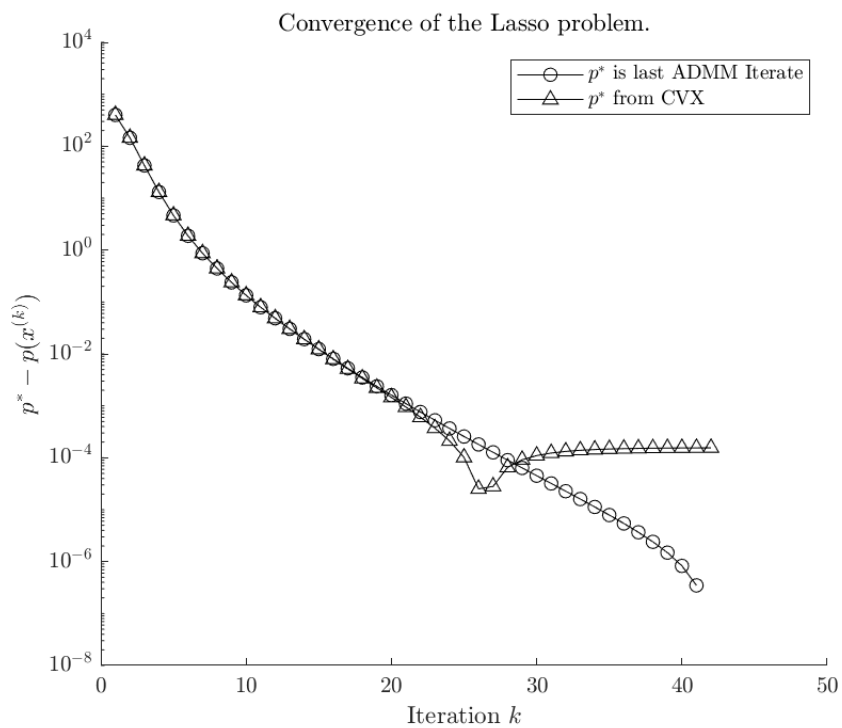


Figure 1: Log plot of the convergence of ADMM, with $\lambda = 1$ and $\rho = 1$. Note, I include two difference scenarios for p^*: (a) the last ADMM iterate and (b) the CVX solution. The dip upwards in the CVX errors is when the ADMM solution produces a smaller solution; and thus we see ADMM defeating CVX in accuracy, and by *far* in runtime (`1.57`s vs `.06`s).

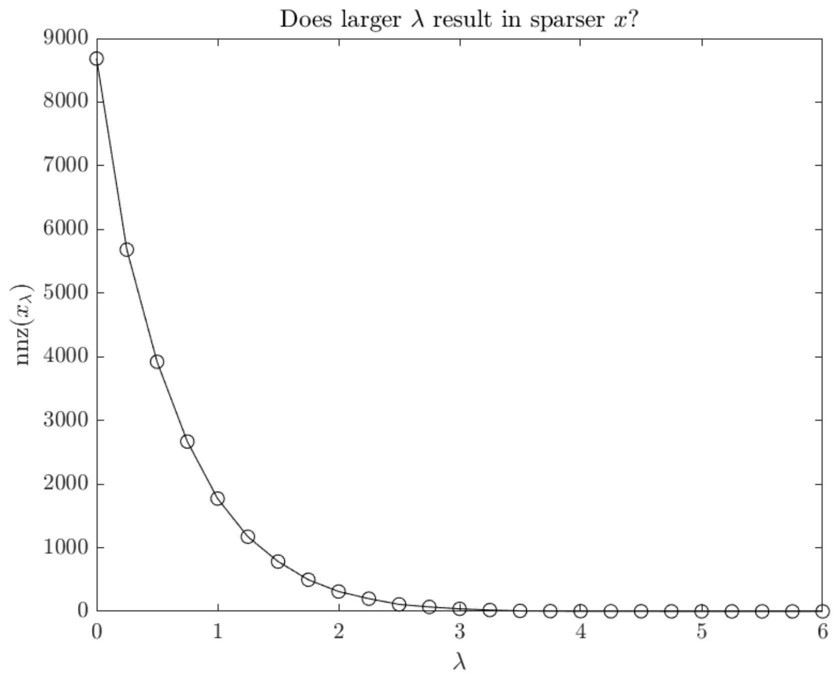


Figure 2: Indeed, as we suspect, increasing λ makes the vectors become sparser. What was surprising to me was how small λ could be before the algorithm decided that the penalty of $\|b\|_2^2/2$ wasn't difficulty to bear. Note: Initializing everything to the zero vector was important, as otherwise you have to check for values within ε of 0, rather than just running $\texttt{nnz}(x)$.

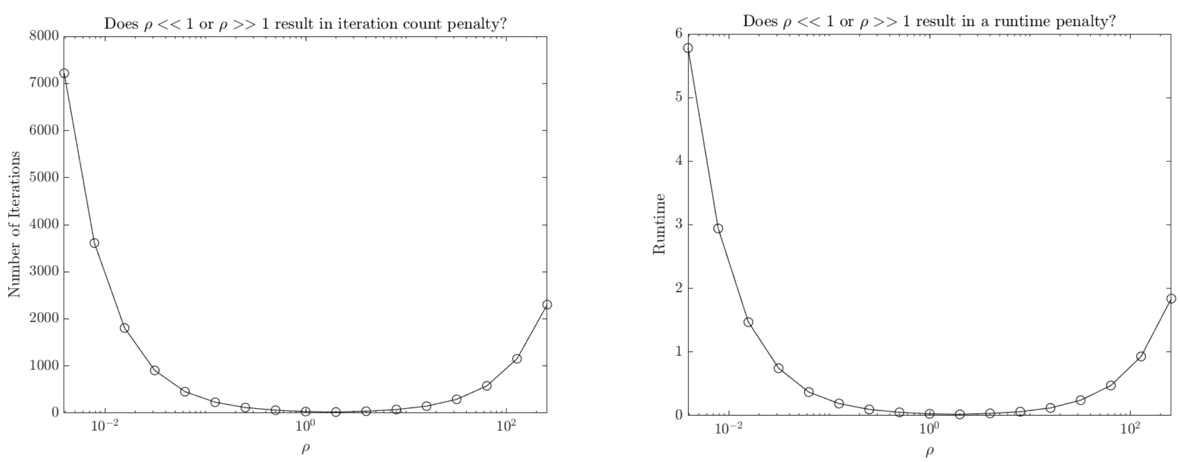
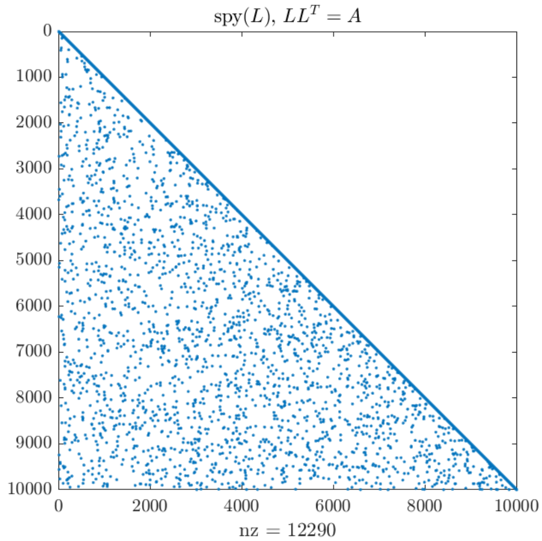


Figure 3: Interesting results here; making ρ too large or too small significantly impacts the number of iterations, and therefore the runtime of ADMM. Beautiful graphs of runtime/iteration vs ρ are produced above; they mirror each other as expected. This is likely due to the factor $\|\rho u\|$ in the computation of $\varepsilon_{\text{dual}}$.



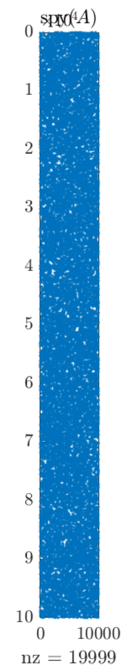


Figure 4: If it weren't for the `nnz` function telling me that A is in fact sparse, it would be hard to tell by the graph. Interestingly, although L is sparse, it doesn't have a special pattern; just lower triangular filled in (not densely).

```
function [x_star, p_star, p_interm] = lasso_admm(A, b, lambda, rho, maxit, tol)
%%% lasso_admm.m
%% Description:
% Solve the Lasso problem:
%   Minimize (1/2) || Ax - b ||_2^2 + \lambda ||x||_1
% using the ADMM method.
%% Input:
% A: M x N matrix.
% b: N x 1 matrix.
% lambda: real number > 0.
% rho: real > 0, Augmented Lagrangian penalty parameter.
%      Controls differentiability of augmented Lagrangian dual function.
% maxit: Maximimum number of ADMM Iterations.
% tol: 1x2 matrix where:
%        tol(1) = absolute tolerance for ADMM stopping criteria.
%        tol(2) = relative tolerance for ADMM stopping critera (~10^-4).
%% Output:
% x: Computed argmin to the Lasso problem.
% p_star: Computed minimum to the Lasso problem.
% p_interm: p_interm(k) computed minimum to Lasso problem at iteration k.

  % Setup Lasso problem + Iteration variables
  [M, N] = size(A);
  S = @(a) max(a-lambda/rho, 0) - max(-a-lambda/rho, 0);
  x = zeros(N,1); z = zeros(N,1); u = zeros(N,1); r = 0; s = 0;
  p_interm = zeros(N,1); x_star = 0; p_star = 0;

  % Precompute expensive operations
  L = chol(A'*A + rho*speye(N), 'lower'); %try iterative solve + preconditioner?
  ATb = A'*b;

  % ADMM Iterations

  for k=1:maxit
    % ADMM Iterate + primal and dual residual computation
    x = L' \ ( L \ ( ATb + rho*(z-u) ) );
    z_new = S(x + u);
    u =  x + u - z_new;

    r = norm(x - z_new);
    s = norm( -rho*(z_new - z) );

    z = z_new;

```

```
    % Compute output values
    p_x = (1/2)*norm(A*x-b,2)^2 + lambda*norm(x,1);
    p_z = (1/2)*norm(A*z-b,2)^2 + lambda*norm(z,1);
    [p_star, idx] = min([p_x p_z]); x_star = (idx == 1)*x + (idx == 2)*z;
    p_interm(k) = p_star;

    % Stopping Criteria (pg.19)
    e_pri  = sqrt(N)*tol(1) + tol(2)*max(norm(x), norm(-z)); %A,-B = I, c = 0
    e_dual = sqrt(N)*tol(1) + tol(2)*norm(rho*u); %y = (1/rho)*y (pg.15)
    if (r <= e_pri) && (s <= e_dual) %equation (3.12)
      break
    end
  end
  p_interm = p_interm(1:k);
end
```

```
function x = lasso_cvx(A, b, lambda)
%%% lasso_cvx.m
%% Description:
% Solve the Lasso problem:
%   Minimize (1/2) || Ax - b ||_2^2 + \lambda ||x||_1
% using CVX.
%% Input:
% A: M x N matrix.
% b: N x 1 matrix.
% lambda: real number > 0.
%% Output:
% x: Computed argmin to the Lasso problem.
  [~,M] = size(A);
  cvx_begin quiet
    variable x(M)
    minimize( 0.5*square_pos( norm(A*x-b,2) ) + lambda*norm(x,1) )
  cvx_end
end
```

```
%%% large_lasso.m
%% Experiments on the parameters of the ADMM method
clf; close all;
% Problem Setup
M = 100000; N = 10000; density = 2/M;
A = sprand(M,N,density); b = randn(M,1);

% Lambda experiment: Does x get sparser?
lambda = (0:0.25:6)';
nnz_lambda = zeros(size(lambda));;
for k = 1:length(lambda)
  l = lambda(k);
  [x, ~, ~] = lasso_admm(A, b, l, 1, 1000, [1e-7 1e-4]);
  fprintf('With lambda = %.2f we receive nnz(x) = %d\n', l, nnz(x));
  nnz_lambda(k) = nnz(x);
  %x = lasso_cvx(A, b, l);
  %fprintf('With lambda = %.2f we receive nnz(x) = %d\n', l, N-sum(abs(x)<1e-7));
end %Yes, around l = 5~ we have x = 0.
figure();
plot(lambda, nnz_lambda, 'k-o');
title('Does larger $\lambda$ result in sparser $x$?')
xlabel('$\lambda$'); ylabel('nnz($x_\lambda$)');

% Rho experiment: What effect does rho have on the method?
lambda = 2.0;
rho = 2.^(-8 : 8)';
timings = size(length(rho));
for k=1:length(rho)
  tic; [~,~,~] = lasso_admm(A,b,lambda,rho(k),10000,[1e-7 1e-4]); t = toc;
  timings(k) = t;
  fprintf('With rho = %.2f has runtime t=%.3f\n ', rho(k), t);
end %As rho << 1 or rho >> 1, the runtime suffers.
figure();
plot(rho, timings, 'k-o');
title('Does $\rho << 1$ or $\rho >> 1$ result in a runtime penalty?')
xlabel('$\rho$'); ylabel('Runtime');
set(gca, 'xscale', 'log');

% Rho experiment: What effect does rho have on the method?
num_its = size(length(rho));
for k=1:length(rho)
  tic; [~,~,its] = lasso_admm(A,b,lambda,rho(k),10000,[1e-7 1e-4]); t = toc;
  num_its(k) = length(its);
  fprintf('With rho = %.2f numits = %d\n ', rho(k), num_its(k));
```

```
end %As rho << 1 or rho >> 1, the number of iterations increases suffers.
figure();
plot(rho, num_its, 'k-o');
title('Does $\rho << 1$ or $\rho >> 1$ result in iteration count penalty?')
xlabel('$\rho$'); ylabel('Number of Iterations');
set(gca, 'xscale', 'log');
```

```
%%% convergence_lasso.m
%% Sanity checks on the convergence of ADMM
clf; close all;
% Problem Setup
M = 100000; N = 10000; density = 2/M;
A = sprand(M,N,density); b = randn(M,1);
lambda = 1.0;
obj = @(x) norm(A*x-b,2).^2/2 + lambda*norm(x,1);

x_star = lasso_cvx(A,b,lambda); p_cvxstar = obj(x_star);
[~,p_star,p_computed] = lasso_admm(A,b,lambda,1,100,[1e-8 1e-5]);

figure();
hold on;
plot(1:length(p_computed), p_computed-p_star, 'k-o');
plot(1:length(p_computed), abs(p_computed-p_cvxstar), 'k-^');
hold off;
title('Convergence of the Lasso problem.');
ylabel('$p^* - p(x^{(k)})$'); xlabel('Iteration $k$');
set(gca, 'yscale', 'log');
```