OLS with $\ell_1$ and $\ell_2$ regularization

CEE 629. System Identification
Duke University, Fall 2017

$\ell_1$ regularization

- The $\ell_1$ norm of a vector $v \in \mathbb{R}^n$ is given by $||v||_1 = \sum |v_i|$
  The gradient of $||v||_1$ is not defined if an element of $v$ is zero.

- In $\ell_1$ regularization, the objective $J(a) = ||y - f(y;a)||_2^2$ is penalized with a term $\alpha||a||_1$, where $\alpha$ is called the regularization parameter, or the penalty factor.

- The effect of $\ell_1$ regularization is to force some of the model parameters, $a_i$, to zero (exactly). If the parameters are coefficients for bases of the model, then $\ell_1$ regularization is a means to remove un-important bases of the model. It is a form of model reduction

- It is applicable to linear and nonlinear models.

- It has similar but not identical effects as $\ell_2$ regularization and truncated SVD solutions

- a.k.a. LASSO, Compressed Sensing, Support Vector Machine

But $\ell_1$ is not smooth at the optimum point.
What can one say about necessary conditions for optimality?

$\ell_1$ regularization can be recast as a Quadratic Program (QP)

$$ J(a) = ||y - Xa||_2^2 + \alpha||a||_1 = ||y - Xa||_2^2 + \alpha \sum_i |a_i| \tag{1} $$

$$ \Leftrightarrow $$

$$ a_i \equiv p_i - q_i \tag{2} $$

$$ |a_i| \equiv p_i + q_i \tag{3} $$

$$ J(p,q) = ||y - Xp + Xq||_2^2 + \alpha \sum_i (p_i + q_i) \tag{4} $$

such that $p_i \geq 0$ and $q_i \geq 0 \ \forall \ i$
Five conditions to show that $\ell_1$ is a QP

1. $p_i - q_i = a_i$
2. $p_i + q_i = |a_i|$
3. $p_i \geq 0 \quad q_i \geq 0$
4. $a_i > 0 \Rightarrow p_i = a_i, \quad q_i = 0$
   $a_i < 0 \Rightarrow p_i = 0, \quad q_i = -a_i$
   $a_i = 0 \Rightarrow p_i = 0, \quad q_i = 0$
5. $\min(p_i + q_i) = a_i$

Solving the $\ell_1$ regularization QP

$$J(p, q) = ||y - Xp + Xq||_2^2 + \alpha \sum_i (p_i + q_i)$$
such that $p_i \geq 0$ and $q_i \geq 0 \quad \forall \ i$

But which elements of $p_i$ and which elements of $q_i$ should be set to zero?
... (the “active” constraints)
This question does not have an answer in closed form.
So we solve the problem incrementally.

Incremental formulation

$$p^{(k+1)} = p^{(k)} + u^{(k)}, \quad p^{(k)} + u^{(k)} \geq 0 \quad (5)$$
$$q^{(k+1)} = q^{(k)} + v^{(k)}, \quad q^{(k)} + v^{(k)} \geq 0 \quad (6)$$

Lagrangian

$$L(p, q, \mu, \nu) = J(p, q) + \mu^{(k)}_{i \in A_u} (p^{(k)} + u^{(k)})_{i \in A_u} + \nu^{(k)}_{i \in A_v} (q^{(k)} + v^{(k)})_{i \in A_v}$$

• $k$ is the iteration number
• $A_u$ is the set of active equality constraints $p_i = 0$
• $A_v$ is the set of active equality constraints $q_i = 0$
• $\mu^{(k)}_{i \in A_u}$ is the set of Lagrange multipliers for the active constraint set $A_u$
• $\nu^{(k)}_{i \in A_v}$ is the set of Lagrange multipliers for the active constraint set $A_v$
Minimize the Lagrangian w.r.t. \((p, q)\),
Maximize the Lagrangian w.r.t. \((\mu, \nu)\)

\[
L = y^T y - 2(p + u)^T X^T y + p^T \alpha \\
+ 2(q + v)^T X^T y + q^T \alpha \\
+ (p + u)^T X^T (p + u) \\
- 2(q + v)^T X^T (p + u) \\
+ \mu^T_{i \in A_u} (p + u)_{i \in A_u} \\
+ \nu^T_{i \in A_v} (q + v)_{i \in A_v}
\]

The necessary conditions for optimality ...

\[
\frac{\partial L}{\partial u} = 0 : -2X^T y + \alpha + \mu_{i \in A_u} + 2X^T X p + 2X^T X u - 2X^T X q - 2X^T X v = 0
\]

\[
\frac{\partial L}{\partial v} = 0 : +2X^T y + \alpha + \nu_{i \in A_v} - 2X^T X p - 2X^T X u + 2X^T X q + 2X^T X v = 0
\]

\[
\frac{\partial L}{\partial \mu_{i \in A_u}} = 0 : (p + u)_{i \in A_u} = 0
\]

\[
\frac{\partial L}{\partial \nu_{i \in A_v}} = 0 : (q + v)_{i \in A_v} = 0
\]

... are linear in our unknowns \(u, v, \mu, \nu\) ...

\[
\begin{bmatrix}
2X^T X & -2X^T X & I^{T}_{i \in A_u} \\
-2X^T X & 2X^T X & I^{T}_{i \in A_v} \\
I_{i \in A_u} & I_{i \in A_v}
\end{bmatrix}
\begin{bmatrix}
u \\
\mu \\
\nu
\end{bmatrix}
= 
\begin{bmatrix}
2X^T y - 2X^T X p + 2X^T X q - \alpha \\
-2X^T y + 2X^T X p - 2X^T X q - \alpha \\
-p_{i \in A_u} \\
-q_{i \in A_v}
\end{bmatrix}
\]

The number of columns of the matrix \(I_{i \in A_u}\) is the number of model parameters. The number of rows of \(I_{i \in A_u}\) is the number of active constraints on \(p\). If index \(i\) is the element \(j\) in the set of active constraints, \(A_u\), then the \(i, j\) element of \(I_{i \in A_u}\) equals 1.
A trick to block an incremental step from going negative.

Define the index $j$ s.t. $p_j + u_j = \min(p + u)$ ... ($p_j \geq 0$)

If $p_j + u_j < 0$, then define $\delta_u = -p_j / u_j$ ($\delta_u > 0$)
and adjust the increment by the factor $\delta_u$

$$p^{(k+1)} = p^{(k)} + \delta_u u^{(k)}$$

Likewise ...

Define the index $j$ s.t. $q_j + v_j = \min(q + v)$ ... ($q_j \geq 0$)

If $q_j + v_j < 0$, then define $\delta_v = -q_j / v_j$ ($\delta_v > 0$)
and adjust the increment by the factor $\delta_v$

$$q^{(k+1)} = q^{(k)} + \delta_v v^{(k)}$$

Implementation in L1_fit.m
OLS with $\ell_1$ and $\ell_2$ regularization

Example 1, 2-term power polynomial

$$\hat{y}(x; a_1, a_2) = a_1 x + a_2 x^2$$

$$y = \hat{y} + \eta \quad \sigma_\eta = 0.5$$

$a_1 = 4, a_2 = 8$

OLS criterion

$$\min_a \| y - Xa \|_2^2$$

$\ell_1$ regularization criterion

$$\min_a \| y - Xa \|_2^2 + \alpha \|a\|_1$$

$\ell_2$ regularization criterion

$$\min_a \| y - Xa \|_2^2 + \beta \|a\|_2$$

---

**Graphs:**

- **Left graph:**
  - True: Black line
  - Measurement: Red points
  - OLS: Green line
  - $L_2, \beta = 1$: Blue line
  - $L_1, \alpha = 20$: Red line

- **Right graph:**
  - True: Black line
  - Measurement: Red points
  - OLS: Green line
  - $L_2, \beta = 70$: Blue line
  - $L_1, \alpha = 70$: Red line

**Axes:**

- $a_1$ on the x-axis
- $a_2$ on the y-axis

**Labels:**

- OLS
- $L_1$
- $L_2$

---

**Additional Information:**

- $\beta = 1.0$
- $\alpha = 20.0$
- $\beta = 70.0$
- $\alpha = 70.0$
Example 2, 100-term Prony Series

A Prony series a model for the behavior dynamic system expressed in terms of a convolution of the system forcing with a kernel, in which the kernel is a series of exponentials.

\[ y(t) = \int_0^t h(r)u(t-r) \, dr \]

where, for example, the kernel can be parameterized as

\[ h(r) = k_0 + \sum_{i=1}^n k_i \exp[-r/\tau_i] \]

Prony series are linear in the coefficients \( k_i \) but are highly nonlinear in the time constants \( \tau_i \).

In the frequency domain, the complex modulus is obtained from the Fourier transform of the convolution kernel,

\[ \hat{H}(\omega) = k_0 + \sum_{i=1}^n k_i \frac{i\omega\tau_i}{1 + i\omega\tau_i} \]

The real part of \( \hat{H}(\omega) \) is called the storage modulus and represents the energy-conserving elasticity of the system. The imaginary part of \( \hat{H}(\omega) \) is called the loss modulus and relates to the energy dissipation aspects. For linear visco-elastic materials the loss modulus approaches zero as \( \omega \) approaches zero.

Using \( \ell_1 \) regularization, a Prony series with very large set of time constants can be fit to complex-modulus data. By tuning the \( \ell_1 \) penalty factor \( \alpha \) to be large enough, a large number of coefficients in the Prony series will be forced to zero. Since negative values of the coefficients are not permissible, this problem is a perfect candidate for \( \ell_1 \) regularization.

The system model is then given by

\[ \hat{H}(\omega; k) = T(\tau)k \]

where the \( i \)-th column of the matrix \( T(\tau) \) is \( i\omega\tau_i/(1 + i\omega\tau_i) \) and the vector \( k \) contains the \( n + 1 \) coefficients \( k_0, ..., k_n \).

The regularized objective function is

\[ J(k) = ||H(\omega) - T(\tau)k||_2^2 + \alpha \sum_i k \quad \text{s.t. } k \geq 0 \]

This is a special (more simple) case of the \( \ell_1 \) objective analyzed in the previous section. The solution is implemented in PronyFit.m.
OLS with $\ell_1$ and $\ell_2$ regularization
% L1_reg_example.m
% HPG, Duke Univ. 2013–09, 2017–09

% generate some data (x,y), noisy in y
m = 100; % number of measurement points
x = [1:m]’/m; % independent variables – noise-free
a = [4 8 ’]; % "true" parameter coefficients
n = length(a);
X = zeros(m,n);
for i=1:n
    X(:,i) = x.^i;
end
[eigvec,eigval] = eig(X’*X); % principal directions of parameter covariance
y = X*a;

alpha_set = [20.0 70.0 ]; % for L1 regularization
beta_set = [1.0 70.0 ]; % for L2 regularization
N_MCS = 100;
N_R = length(alpha_set);
a_hat_ols_rec = zeros(n,N_MCS);
a_hat_alpha_rec = zeros(n,N_R,N_MCS);
a_hat_beta_rec = zeros(n,N_R,N_MCS);

for j=1:N_MCS;
    % ... start simulation loop
    y_dat = y + 0.5*randn(m,1); % add noise to true data
    a_hat_ols = X \ y_dat; % ordinary least squares
    y_hat_ols = X*a_hat_ols;
a_hat_ols_rec(:,j) = a_hat_ols;
    for k=1:N_R
        alpha = alpha_set(k);
        beta = beta_set(k);
        a_hat_alpha = L1_fit(X, y_dat, alpha ); % L_1 regularization
        y_hat_alpha = X * a_hat_alpha;
a_hat_alpha_rec(:,j,k) = a_hat_alpha;
        a_hat_beta = [X’*X + beta*eye(n) ] \ X’*y_dat; % L_2 regularization
        y_hat_beta = X*a_hat_beta;
a_hat_beta_rec(:,j,k) = a_hat_beta;
    end
figure(1)
clf
plot(x,y,’-k’, x,y_dat,’ok’, ... 
x,y_hat_ols,’-k’, x,y_hat_beta,’-b’, x,y_hat_alpha,’-r’ )
legend(’true’,’msmnt’,’OLS’, ... 
sprintf(’L_2’, \beta = %2.0f’,beta), ... 
sprintf(’L_1’, \alpha = %2.0f’,alpha),’location’,’northwest’);
legend(’boxoff’)
drawnow
end
end % ... end simulation loop
function [a, mu, nu, cvg_hst] = L1_fit(X, y, alpha)
% [a, mu, nu, cvg_hst] = L1_fit(X,y, alpha)
% fit model parameters, a, in the model \( \hat{y} = Xa \) to data, y, with
% L1 regularization of the parameters.
% \( J = ||y - Xa||^2_2 + alpha \sum |a| \_1 \)
% \( J = ||y - X(p-q)||^2_2 + alpha \sum |p+q| \_1 \) such that \( p\geq0, q\geq0 \)
% \( a = p-q; \quad |a| = p+q \)
% INPUT: X ... the design matrix (basis of the model), \( m \times n \)
% y ... the vector of data to be fit to the model, \( m \times 1 \)
% alpha ... L1 regularization factor for Prony series coefficients
% OUTPUT: a ... the model coefficients
% mu, nu ... Lagrange multipliers for p and q
% cvg_hst ... convergence history
%
%
% The main idea behind casting L1 as a QP is that the parameter vector \( a \) is replaced by the difference of two vectors \( a = \{p\} - \{q\} \) that are constrained to be non-negative: \( p_i > 0 \) for all \( i \) and \( q_i > 0 \) for all \( i \).
% If \( a_i > 0 \), then \( p_i = a_i \) and \( q_i = 0 \);
% If \( a_i < 0 \), then \( p_i = 0 \) and \( q_i = -a_i \).
% With this constrained re-parameterization, \( |a_i| = p_i + q_i \).
% Note that the dimension of the parameter space doubles, but the KKT equations for the QP are simple and have analytical Hessians and gradients.
%
[m, n] = size(X);
t = 1:m;

XTX = 2*X'*X; Xty = 2*X'*y;

% good initial guess for p and q from non-regularized linear least squares
a = XTX \ Xty;
p = zeros(n,1); p(find(a > 2*eps)) = a(find(a > 2*eps));
q = zeros(n,1); q(find(a < -2*eps)) = -a(find(a < -2*eps));

MaxIter = 19;
cvg_hst = zeros (5*n, MaxIter); % convergence history

for iter = 1:MaxIter
    Au = find(p <= 2*eps);  lp = length(Au); % active set for update u
    Av = find(q <= 2*eps);  lq = length(Av); % active set for update v

    Ip = zeros(lp,n); for i=1:lp, Ip(i,Au(i)) = 1; end % constraint gradient u
    Iq = zeros(lq,n); for i=1:lq, Iq(i,Av(i)) = 1; end % constraint gradient v

    mu = zeros(n,1); nu = zeros(n,1);

    % KKT equations
    XTX = [XTX -XTX Ip zeros(n,1); % dL/du = 0
            -XTX XTX zeros(n,lp) Ip'; % dL/dv = 0
            Ip zeros(lp,n+lp+1q) zeros(lq,lp+1q) ]; % dL/d mu = 0
    zeros(lq,n) lq zeros(lq,lp+1q)]; % dL/d nu = 0

    XTY = [Xty -XTX*p + XTX*q - alpha ; % dL/du = 0
            -XTX + XTX*p - XTX*q - alpha ; % dL/dv = 0
            -p(Au); % dL/d mu = 0
            -q(Av)]; % dL/d nu = 0

    u_v_mu_nu = XTX \ XTY; % solve the system
u = u_v_mu_nu(1:n); % update for p
v = u_v_mu_nu(n+1:2*n); % update for q
mu(Au) = u_v_mu_nu(2*n+1:2*n+lp); % what to use mu for?
nu(Av) = u_v_mu_nu(2*n+lp+1:2*n+lp+lq); % what to use nu for?

% if an element of p+u becomes negative, reduce the length of step u
du = 1;
[p_min, j] = min(p+u);
if p_min < 0
du = -p(j)/u(j);
end

% if an element of q+v becomes negative, reduce the length of step v
dv = 1;
[q_min, j] = min(q+v);
if q_min < 0
dv = -q(j)/v(j);
end

p = p + du*u;
q = q + dv*v;
a = p - q;
cvg_hst(:, iter) = [a; p; q; mu; nu]; % convergence history

% convergence check
if (norm(u) <= norm(p)/1e3 && min(p) > -5*eps &&
norm(v) <= norm(q)/1e3 && min(q) > -5*eps)
break;
end

% figure (101); clf; plot(t,y,'o',t,X*a); drawnow;
end
cvg_hst = cvg_hst(:,1:iter);

% figure (102); clf; plot([1:iter], cvg_hst(1:n,:),'-o'); drawnow; %pause(1);

% ----------------------------------------------- L1_fit

% H.P. Gavin, 2013–10–04
OLS with $\ell_1$ and $\ell_2$ regularization

PronyFit_test.m

```matlab
% --- PronyFit_test.m
% test prony series fitting with L1 regularization
clear all

% simulated noisy data
% a relatively small number of 'true' time lags
tau = 1e-2*[ 0.01 0.1 1 10 100 1000 ];
ko = 1; % static stiffness
k = 0.5 ./ tau.^0.2; % 'true' Prony series coefficients

% number of terms in the Prony series
nID = 97;

tau_id = logspace(-4.5,2,nID); % specified set of time lags, tau
alpha = 0.50; % l_1 regularization parameter

[ko_id,k_id,cvg_hst] = PronyFit(G_dat,f_dat,tau_id,alpha); % do it!

% number of terms in the Prony series
nID = 97;

tau_id = logspace(-4.5,2,nID); % specified set of time lags, tau
alpha = 0.50; % l_1 regularization parameter

[ko_id,k_id,cvg_hst] = PronyFit(G_dat,f_dat,tau_id,alpha); % do it!
```

CC BY-NC-ND January 19, 2019, H.P. Gavin
function [ko, k, cvg_hst] = PronyFit(G_dat, f_dat, tau, alpha)

% INPUT: G_dat ... complex modulus data Mx1 complex vector
% f_dat ... frequencies where the measured frequency response data is
% evaluated (Hz), (f)>0 Mx1 real vector
% tau ... specified set of relaxation times, Nx1 real vector
% alpha ... l1 regularization factor for Prony series coefficients

% OUTPUT: ko, k ... Prony series coefficients
cvg_hst ... convergence history

i = sqrt(-1.0);

m = length(f_dat); % number of data points
w = 2*pi*f_dat; % one−sided frequency

T = [ones(m,1), i*w*tau'./(i*w*tau' + 1.0)]; % design matrix

% plot the basis functions (columns of T) with single−sided log−scale frequencies
figure(101)
semilogx(f_dat, real(T), '-r', f_dat, imag(T), '-b')
ylabel('basis functions T_k (\omega) = i \omega \tau_k / (i \omega \tau_k + 1) ')
xlabel('frequency , f, Hz ')
axis([0.5*min(f_dat) 1.5*max(f_dat) -0.05 1.05])

[m,n] = size(T); % m = number of data points; n = number of parameters

TtT = 2*real(T'*T); % taking real part is like adding complex conjugate
TtG = 2*real(T'*G_dat);

k = TtT \ TtG; % O.L.S. fit is a better initial guess, even though infeasible , k<0

MaxIter = 20; % usually enough
cvg_hst = zeros(2*n,MaxIter); % convergence of coefficients and multipliers

for iter = 1:MaxIter % —— Start Main Loop

A = find(k < 2*eps); l = length(A); % active set for update constraints
% Ina is the constraint gradient w.r.t. the step vector h
Ina = zeros(1,n); for i=1:l, Ina(i,A(i)) = 1; end

% KKT equations
lambda = zeros(n,1); % Lagrange multiplier
XTX = [TtT Ina';
       Ina zeros(1,1)]; % dL / dh

XTY = [-TtT*k + TtG - alpha ;
       -k(A)]; % dL / d lambda

h_lambda = XTX \ XTY; % solve the system

h = h_lambda(1:n); % the step
lambda(A) = h_lambda(n+1:end); % the non−zero multipliers

% if an element of k+h becomes negative , reduce the length of step h to dh*h
if h_test_min < 0
    dh = -k(idx)/h(idx);
end
\[ k = (k + dh \cdot h); \quad \% \text{update the Prony coefficients} \]

\[ \text{figure(102)} \quad \% \text{plot meas points and fit} \]

\[ G_{\text{hat}} = T \cdot k; \]

\[ \text{clf} \]

\[ \text{subplot(211)} \]

\[ \text{semilogx}(w, \text{real}(G_{\text{dat}}), 'or', w, \text{real}(G_{\text{hat}}), '-k') \]

\[ \text{ylabel}(\text{'storage modulus, } G'(\omega))' \]

\[ \text{legend}('G'(\omega) \text{ meas.}','G'(\omega) \text{ fit}', 'location','northwest') \]

\[ \text{subplot(212)} \]

\[ \text{semilogx}(w, \text{imag}(G_{\text{dat}}), 'ob', w, \text{imag}(G_{\text{hat}}), '-k') \]

\[ \text{xlabel}(\text{'\omega, rad/s'}) \]

\[ \text{legend}('G''(\omega) \text{ meas.}','G''(\omega) \text{ fit}', 'location','northwest') \]

\[ \text{drawnow} \]

\[ \text{cvg_hst(:, iter)} = [k; lambda] \quad \% \text{convergence history} \]

\[ \text{if (norm(h) < norm(k)/1e3 && min(k) > -5*eps)} \quad \% \text{convergence check} \]

\[ \text{break}; \]

\[ \text{end} \]

\[ \% \text{--- End Main Loop} \]

\[ \text{cvg_hst = cvg_hst(:,1:iter)}; \]

\[ \% \text{asymptotic standard errors of the Prony coefficients} ??? \]

\[ k_{\text{std}}_\text{err} = \text{sqrt}(\text{norm}(G_{\text{dat}}-T \cdot k)/(m-n+1) * \text{diag}(real(inv(TTT)))); \]

\[ \text{ko} = k(1); \]

\[ k = k(2:n); \]

\[ \% \quad \text{PRONY FIT} \]

\[ \% \quad \text{HP Gavin, 2013-10-04} \]