**LASSO 结合原理跟代码**

在晚上下了一个LASSO 的程序，准备结合 基于LASSO的人脸识别算法，结合着看。

clear; close all; clc;

�dpath('lib');

使用的数据库是diabetes

load diabetes

X = diabetes.x;

X为442 行，10 列

X = normalize(X);

这一步的意思是把X的每一列变为均值为0，长度为1 的列

y = diabetes.y;

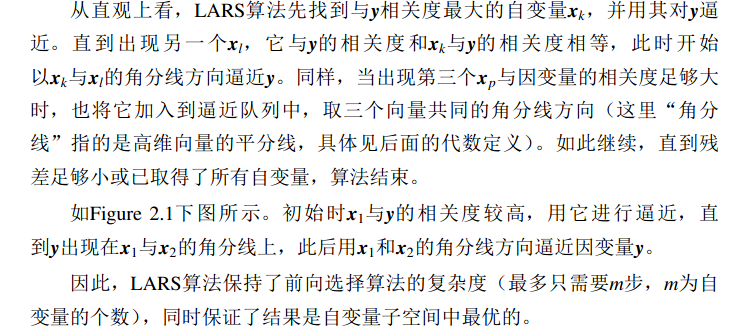
y = center(y);

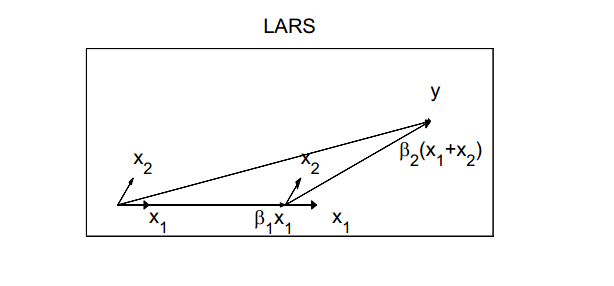
Y 中心化

[n p] = size(X);

b1 = lars(X, y, 'lasso', 0, 0, [], 1);

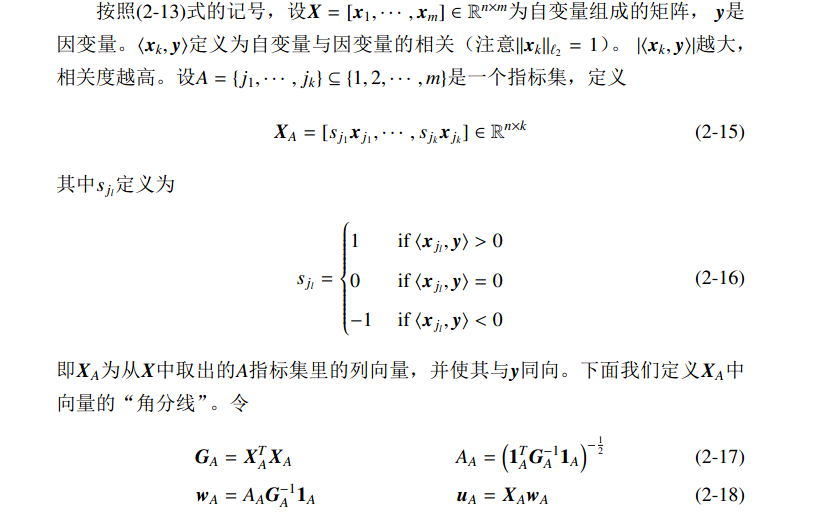
下面是清华大学的那篇文章里面写的： 我自己觉得写的比较清楚。适合像我这样的初学者。

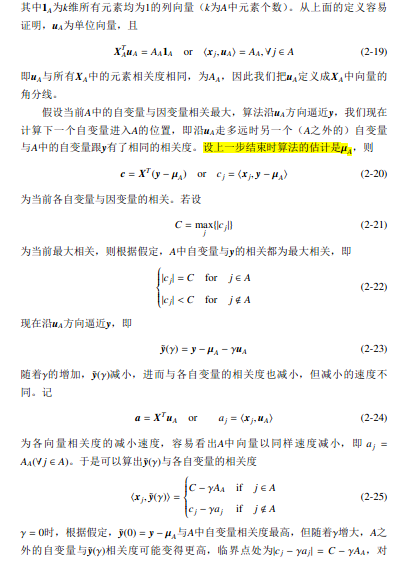




从这个图中我们可以看到假设开始时,y与X1的相关度最高，就用X1，一直等到某一个点，X2与Y的匹配度与X1 一样高，则，采用等角线，依次类推。

此处有一个问题，假如,Y 的维度与X 的维度是一样的，Y的行数与X的函数相同，如果不相同的情况下，怎么办，这样我们就不能通过就内积来决定是不是相关了。





function beta = lars(X, y, method, stop, useGram, Gram, trace）

% LARS The LARS algorithm for performing LAR or LASSO.

% BETA = LARS(X, Y) performs least angle regression on the variables in

% X to approximate the response Y. Variables X are assumed to be

% normalized (zero mean, unit length), the response Y is assumed to be

% centered.

% BETA = LARS(X, Y, METHOD), where METHOD is either 'LARS' or 'LARS'

% determines whether least angle regression or lasso regression should

% be performed.

% BETA = LARS(X, Y, METHOD, STOP) with nonzero STOP will perform least

% angle or lasso regression with early stopping. If STOP is negative,

% STOP is an integer that determines the desired number of variables. If

% STOP is positive, it corresponds to an upper bound on the L1-norm of

% the BETA coefficients.

% BETA = LARS(X, Y, METHOD, STOP, USEGRAM) specifies whether the Gram

% matrix X'X should be calculated (USEGRAM = 1) or not (USEGRAM = 0).

% Calculation of the Gram matrix is suitable for low-dimensional

% problems. By default, the Gram matrix is calculated.

% BETA = LARS(X, Y, METHOD, STOP, USEGRAM, GRAM) makes it possible to

% supply a pre-computed Gram matrix. Set USEGRAM to 1 to enable. If no

% Gram matrix is available, exclude argument or set GRAM = [].

% BETA = LARS(X, Y, METHOD, STOP, USEGRAM, GRAM, TRACE) with nonzero

% TRACE will print the adding and subtracting of variables as all

% LARS/lasso solutions are found.

% Returns BETA where each row contains the predictor coefficients of

% one iteration. A suitable row is chosen using e.g. cross-validation,

% possibly including interpolation to achieve sub-iteration accuracy.

%

% Author: Karl Skoglund, IMM, DTU, kas@imm.dtu.dk

% Reference: 'Least Angle Regression' by Bradley Efron et al, 2003.

%% Input checking

% Set default values.

if nargin < 7

trace = 0;

end

if nargin < 6

Gram = [];

end

if nargin < 5

useGram = 1;

end

if nargin < 4

stop = 0;

end

if nargin < 3

method = 'lars';

end

if strcmpi(method, 'lasso')

lasso = 1;

else

lasso = 0;

end

%% LARS variable setup

[n p] = size(X);

nvars = min(n-1,p); %

maxk = 8\*nvars; % Maximum number of iterations

if stop == 0

beta = zeros(2\*nvars, p);

此处beta 为 20 行10 列的矩阵。

elseif stop < 0

beta = zeros(2\*round(-stop), p);

else

beta = zeros(100, p);

end

mu = zeros(n, 1); % current "position" as LARS travels towards lsq solution

mu 为442 行 1 列的0 矩阵。

I = 1:p; % inactive set

开始认为10列全不是积极集

A = []; % active set

只有0 个积极集

% Calculate Gram matrix if necessary

if isempty(Gram) && useGram

Gram = X'\*X; % Precomputation of the Gram matrix. Fast but memory consuming.

end

if ~useGram

R = []; % Cholesky factorization R'R = X'X where R is upper triangular

当R 为上三角时，Cholesky factorization

end

lassocond = 0; % LASSO condition boolean

stopcond = 0; % Early stopping condition boolean

k = 0; % Iteration count

vars = 0; % Current number of variables 目前变量的个数。

好像不是迭代次数的相加。因为好像过了十次还是12次之后还是继续循环，没有停止的意思。

if trace

disp(sprintf('Step\tAdded\tDropped\t\tActive set size'));

end

%% LARS main loop maxk 为最大的迭代次数。

while vars < nvars && ~stopcond && k < maxk

k = k + 1; k 才为迭代次数

c = X'\*(y - mu); 看来MU 应该是逼近y 的向量。

[C j] = max(abs(c(I)));

j = I(j); 找出来那个与Y 最相关。

if ~lassocond % if a variable has been dropped, do one iteration with this configuration (don't add new one right away)

if ~useGram

R = cholinsert(R,X(:,j),X(:,A));

这条语句的意思是不是把最相关的向量加入到积极集里面去。

好像不是，因为运行完之后R 为1.000 是X(:,j)\*X(:,j) =1

end

A = [A j]; 把最相关的加入到积极集

I(I == j) = []; 把进入积极集的从非积极集中踢出来。

vars = vars + 1; 目前的积极集中的变量个数

if trace

disp(sprintf('%d\t\t%d\t\t\t\t\t%d', k, j, vars));

end

end

s = sign(c(A)); % get the signs of the correlations

if useGram

S = s\*ones(1,vars);

GA1 = inv(Gram(A,A).\*S'.\*S)\*ones(vars,1);

AA = 1/sqrt(sum(GA1));

w = AA\*GA1.\*s; % weights applied to each active variable to get equiangular direction

else

GA1 = R\(R'\s);

AA = 1/sqrt(sum(GA1.\*s));

w = AA\*GA1;

w 为固定的计算方式，这三行都为计算w

end

u = X(:,A)\*w; % equiangular direction (unit vector)

if vars == nvars % if all variables active, go all the way to the lsq solution

gamma = C/AA; 如果所有的变量都是积极集，则变成一个最小二乘问题。

else

a = X'\*u; % correlation between each variable and eqiangular vector

temp = [(C - c(I))./(AA - a(I)); (C + c(I))./(AA + a(I))];

gamma = min([temp(temp > 0); C/AA]); 计算在这个方向上走多远

gamma的取值1.temp(temp > 0)（这一步是计算的啥，没看懂）; 2.C/AA

end

% LASSO modification

if lasso

lassocond = 0;

temp = -beta(k,A)./w';

[gamma\_tilde] = min([temp(temp > 0) gamma]);

j = find(temp == gamma\_tilde);

if gamma\_tilde < gamma,

gamma = gamma\_tilde;

lassocond = 1;

end

end

mu = mu + gamma\*u; 逼近Y 的向量

if size(beta,1) < k+1

beta = [beta; zeros(size(beta,1), p)];

end

beta(k+1,A) = beta(k,A) + gamma\*w';

每个A 中的自变量相应增加 gamma\*w'，这是什么意思。

% Early stopping at specified bound on L1 norm of beta

if stop > 0

t2 = sum(abs(beta(k+1,:)));

if t2 >= stop

t1 = sum(abs(beta(k,:)));

s = (stop - t1)/(t2 - t1); % interpolation factor 0 < s < 1

beta(k+1,:) = beta(k,:) + s\*(beta(k+1,:) - beta(k,:));

stopcond = 1;

end

end

% If LASSO condition satisfied, drop variable from active set

if lassocond == 1

if ~useGram

R = choldelete(R,j);

end

I = [I A(j)];

A(j) = [];

vars = vars - 1;

if trace

disp(sprintf('%d\t\t\t\t%d\t\t\t%d', k, j, vars));

end

end

% Early stopping at specified number of variables

if stop < 0

stopcond = vars >= -stop;

end

end

% trim beta

if size(beta,1) > k+1

beta(k+2:end, :) = [];

end

if k == maxk

disp('LARS warning: Forced exit. Maximum number of iteration reached.');

end

%% Fast Cholesky insert and remove functions

% Updates R in a Cholesky factorization R'R = X'X of a data matrix X. R is

% the current R matrix to be updated. x is a column vector representing the

% variable to be added and X is the data matrix containing the currently

% active variables (not including x).

function R = cholinsert(R, x, X)

diag\_k = x'\*x; % diagonal element k in X'X matrix

if isempty(R)

R = sqrt(diag\_k);

else

col\_k = x'\*X; % elements of column k in X'X matrix

R\_k = R'\col\_k'; % R'R\_k = (X'X)\_k, solve for R\_k

R\_kk = sqrt(diag\_k - R\_k'\*R\_k); % norm(x'x) = norm(R'\*R), find last element by exclusion

R = [R R\_k; [zeros(1,size(R,2)) R\_kk]]; % update R

end

% Deletes a variable from the X'X matrix in a Cholesky factorisation R'R =

% X'X. Returns the downdated R. This function is just a stripped version of

% Matlab's qrdelete.

function R = choldelete(R,j)

R(:,j) = []; % remove column j

n = size(R,2);

for k = j:n

p = k:k+1;

[G,R(p,k)] = planerot(R(p,k)); % remove extra element in column

if k < n

R(p,k+1:n) = G\*R(p,k+1:n); % adjust rest of row

end

end

R(end,:) = []; % remove zero'ed out row

%% To do

%

% There is a modification that turns least angle regression into stagewise

% (epsilon) regression. This has not been implemented.