```
function [beta,status,history] = jacobi(X, y, lambda)
% @brief l1-Regularized Least Squares Solver:
%   l1_ls solves problems of the following form:
%       minimize ||y - X * beta||_2^2 + lambda * ||beta_i||_1,
%   where X and y are problem data and beta is variable (described
 below).
%
% gauss seidel versus jacobi is used
%
% @author Xiaoyun Yuan,
% @date   Oct 15, 2017
%

% set parameters
status = 'false';
max_iter = 200;

% data size
n = size(X, 1);
p = size(X, 2);

% init beta
beta = zeros(p, 1);
obj = norm(y - X * beta, 2) ^ 2 + lambda * norm(beta, 1);
history = zeros(max_iter, 3);

fprintf('\nSolving a problem of size (n=%d, p=%d), with lambda=%.5e
\n',...
            n, p, lambda);
fprintf('------------------------------------------------------------------------
\n');
fprintf('%5s %9s %11s', 'iter','obj', 'reltot');
fprintf('\n');
fprintf('%4d %12.2e %15.5e\n',...
        0, obj, nan);

for iter = 0:max_iter
   % start optimization
   beta_new = beta;
   for para_i = 1:p
       beta_2 = beta;
       beta_2(para_i) = 0;
       z = y - X * beta_2;
       X_i = X(:, para_i);
       % find the best beta_i
       % assume beta_i > 0
       beta_i1 = (z' * X_i - lambda / 2) / norm(X_i, 2) ^ 2;
       if beta_i1 < 0
           beta_i1 = 0;
       end
       beta_i1_ = beta_2;
       beta_i1_(para_i) = beta_i1;
```

```
      obj1 = norm(y - X * beta_i1_, 2) ^ 2 + lambda * norm(beta_i1_,
1);

      beta_i2 = (z' * X_i + lambda / 2) / norm(X_i, 2) ^ 2;
      if beta_i2 > 0
          beta_i2 = 0;
      end
      beta_i2_ = beta_2;
      beta_i2_(para_i) = beta_i2;
      obj2 = norm(y - X * beta_i2_, 2) ^ 2 + lambda * norm(beta_i2_,
1);

      if (obj1 < obj2)
         beta_new(para_i) = beta_i1;
      else
         beta_new(para_i) = beta_i2;
      end
   end
   beta = beta_new;
   % calculate obj
    % calculate current
   obj_new = norm(y - X * beta, 2) ^ 2 + lambda * norm(beta, 1);
   fprintf('%4d %12.5e %15.5e\n',...
        iter + 1, obj_new, abs(obj - obj_new) / obj_new);
   history(iter + 1, :) = [iter, obj, abs(obj - obj_new) / obj_new];
   if (abs(obj - obj_new) / obj_new < 1e-3)
       status = 'Local minimum found!\n';
       break;
   end
   obj = obj_new;
end

history = history(1:iter + 1, :);
```

```
Not enough input arguments.

Error in jacobi (line 18)
n = size(X, 1);
```

Published with MATLAB® R2016a