APPC
TP 3 Component Wise Lasso

Thomas Robert

Component Wise Lasso function

First, we create a function to implement Component-Wise Lasso algorithm.

```
function b = CWLasso(X, y, lambda, b)

p = size(X, 2);

% init iteration variables
JOld = 0;
J = Inf;
i = 0;

% while we don't iterate too much (avoid infinite loop) & the cost varies
while (i < 10000 && abs(JOld - J) > 10e-7)

    % go through all p's
    for pj = randperm(p)

        % Compute bj_MC for j-th component
        x = X(:,pj);
        zInds = setdiff(1:p,pj);
        z = y - X(:,zInds)*b(zInds);

        bjMC = (x'*z)/(x'*x);

        % compute bj
        b(pj) = sign(bjMC)*max(0, abs(bjMC)-lambda/(x'*x));
    end

    % compute J (cost)
    JOld = J;
    J = norm(X*b - y)^2;

    % increment i
    i = i + 1;
end
```

Prepare data

First we prepare the data : we standardize the data and create 2 datasets for learning and testing.

```
close all;
data = load('housing.data');

% make X and y matrices
[n,d] = size(data);
p = d-1;
X = data(:, 1:p);
y = data(:,d);

% standardize feature values and center target
mu_y = mean(y);
y = y - mu_y;
[X, mu, sigma] = standardizeCols(X);

% Split learn and test
```

```
[Xlearn, ylearn, Xtest, ytest] = splitdata(X, y, 0.5);
```

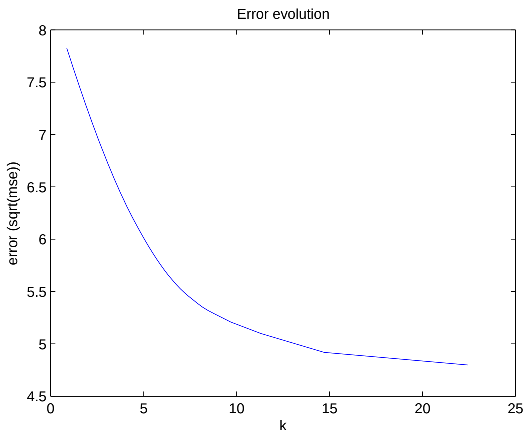
Compute betas

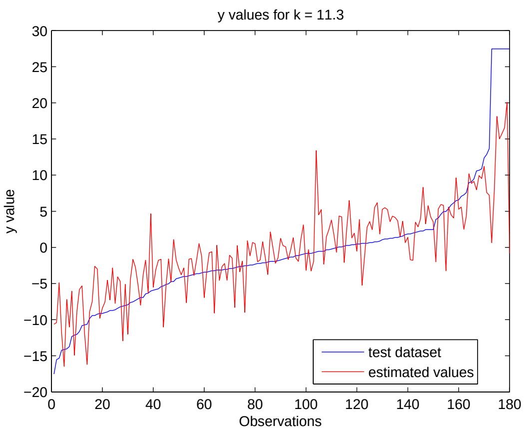
We compute values of β thanks to the function CWLasso that is a component wise implementation of Lasso.

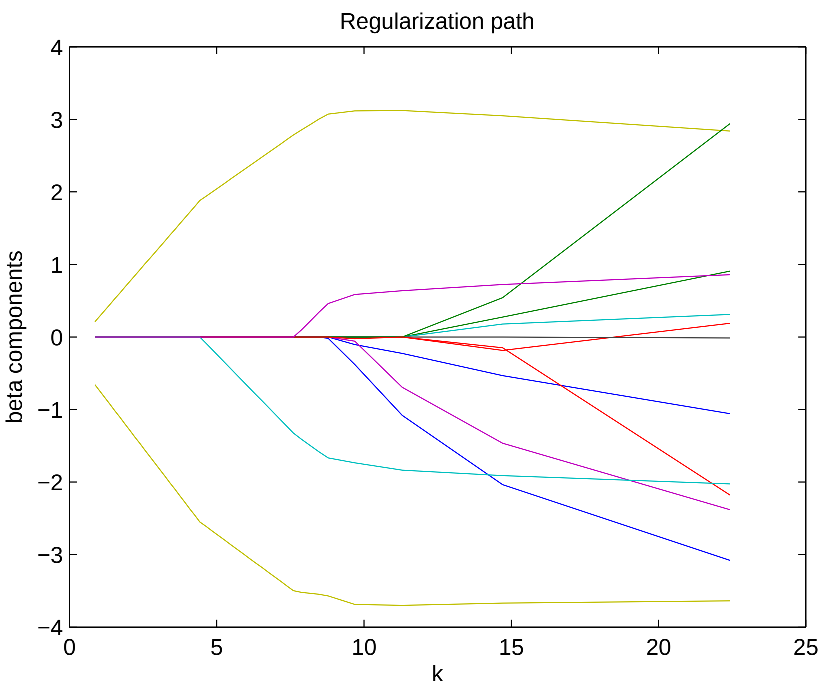
```
Lvals = [0:50:2200];
errors = zeros(length(Lvals), 1);
betas = zeros(length(Lvals), p);
betasCVX = zeros(length(Lvals), p);

for i = 1:length(Lvals)
    L = Lvals(i);
    betas(i,:) = CWLasso(Xlearn, ylearn, L, zeros(p,1));
    ytest_hat = Xtest * betas(i,:)';
    errors(i) = sqrt(mean((ytest - ytest_hat).^2));
end
```

Plot results

We plot the results. We can see that even if MSE still seems to give the best error rate on the test dataset, we already have pretty good results with fewer variables, for example see the chart with $k = 10$.







```
ks = sum(abs(betas'));

% plot error evolution
figure;
plot(ks, errors');
title('Error evolution');
xlabel('k');
ylabel('error (sqrt(mse))');

% plot k = 8
ind = find(ks > 10, 1, 'last');
L = Lvals(ind);
figure;
plot(ytest, 'b');
hold on;
plot(Xtest * betas(ind, :)', 'r');
title(['y values for k = ' num2str(ks(ind))]);
xlabel('Observations');
ylabel('y value');
legend('test dataset', 'estimated values', 'Location', 'Best');

% plot regularization path

figure;
plot(ks, betas');
title('Regularization path');
xlabel('k');
ylabel('beta components');
```

Check results

We compute one of these results with CVX to check if the results are good. We obtain a small error (around 10^{-5}), we can conclude that CW Lasso converge to "standard" Lasso computation.

```
% We check the i = 30th k value
i = 30;
k = ks(i);

% Resolve min problem
cvx_quiet(true);
cvx_begin
    % variables
    variables b(p)

    % objectif
    minimise(1/2 * b'*(Xlearn')*Xlearn*b - ylearn'*Xlearn*b)

    % contraintes
    subject to
        norm(b, 1) <= k
cvx_end

differenceWithCW = norm(betas(i,:)' - b)
```

```
differenceWithCW = 1.9132e-05
```