PQSQ versus LASSO and Elastic net

Contents

[1 Introduction 2](#_Toc456363785)

[2 PQSQ functions 3](#_Toc456363786)

[3 PQSQ regression regularisation 4](#_Toc456363787)

[4 Black hole 5](#_Toc456363788)

[5 Algorithm for PQSQ regression regularisation 5](#_Toc456363789)

[6 Comparison of lasso and PQSQ regularized regression 5](#_Toc456363790)

[7 Comparison of elastic net and PQSQ regularized regression 10](#_Toc456363791)

[8 Test of MatLab lasso function 13](#_Toc456363792)

[9 MatLab implementation 16](#_Toc456363793)

[Reference 19](#_Toc456363794)

# Introduction

Let us consider regression problem with data matrix and response vector , where is row vector which corresponds to th object. We can also write where is column vector which corresponds to values of th attribute. Standard linear regression equation is

|  |  |
| --- | --- |
|  | (1) |

where is vector of regression coefficients and is intercept. The widely used method of regression coefficient search is Ordinal Least Square (OLS). OLS search the minimum of sum of squared deviations:

|  |  |
| --- | --- |
|  | (2) |

The main preference of OLS is transforming of problem (2) to System of Linear Algebraic Equations (SLAE). Indeed, let us calculate derivatives of function with respect to and set it zero.

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

After some algebra we find

|  |  |
| --- | --- |
|  | (5) |
|  | (6) |

where ,, , and .

Linear regression with coefficients which are found from SLAE (5) and (6) has two known drawback [2]: prediction accuracy and low interpretability. Problems with prediction accuracy are sourced by low bias and high variance of estimated by OLS. Big number of predictors reduces interpretability of regression. There are two standard approaches to solve these problems: selection of subset of attributes and ridge regression. The main drawback of selection of subset of attributes is discrete nature of process [2]. In paper [2] the lasso approach is proposed to improve regression coefficients. Elastic net [3] is the method to combine lasso and ridge regression. The main drawback of lasso is low speed of calculation for all methods, include least angle regression algorithm [4].

The function of interest of elastic net is

|  |  |
| --- | --- |
|  | (7) |

We suggest usage of PQSQ functions [5] for regression regularisation. The general PQSQ regularised regression function of interest is

|  |  |
| --- | --- |
|  | (8) |

where is PQSQ function for th regularisation, is weight of th regularisation.

# PQSQ functions

PQSQ function introduced in [5] is function which is defined by set of intervals

|  |  |
| --- | --- |
|  | (9) |

and majorant function :

|  |  |
| --- | --- |
|  | (10) |

where and are coefficients which is defined by equation

|  |  |
| --- | --- |
|  | (11) |

Equation (10) defines unique coefficients and for :

|  |  |
| --- | --- |
|  | (12) |

For the border cases we have

|  |  |
| --- | --- |
|  | (13) |

Examples of PQSQ functions for and are presented in Figure 1. We can see that for function all parabolas which corresponds to three intervals are coincides with majorant function. It means that all PQSQ functions with quadratic majorant function are the same and can be presented as PQSQ function with one interval.

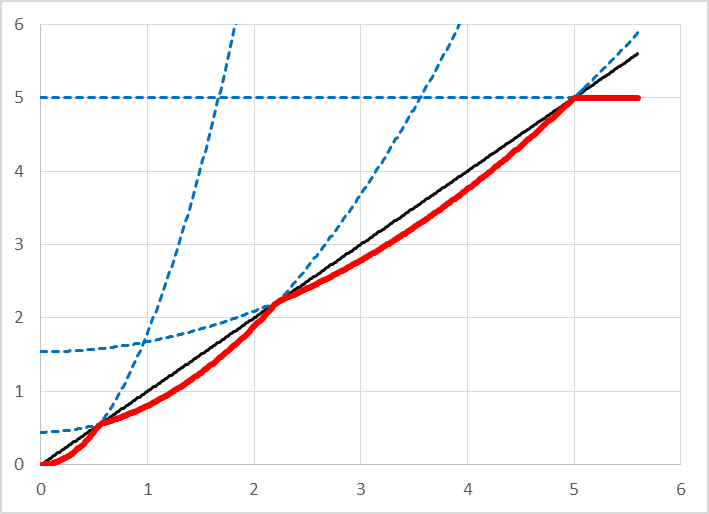
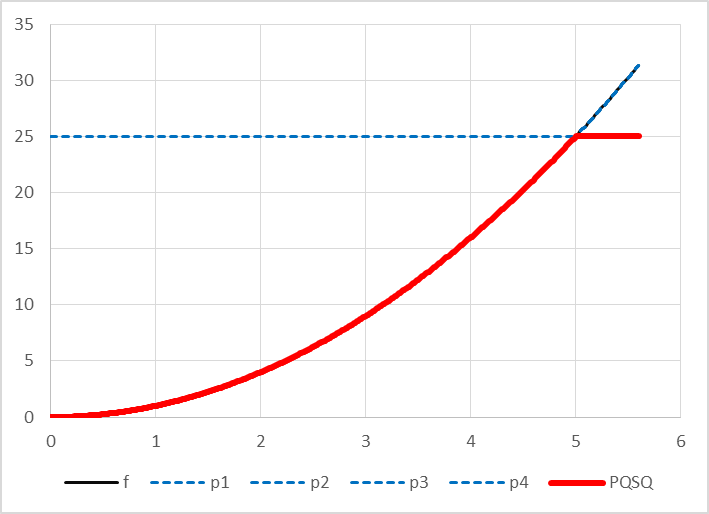
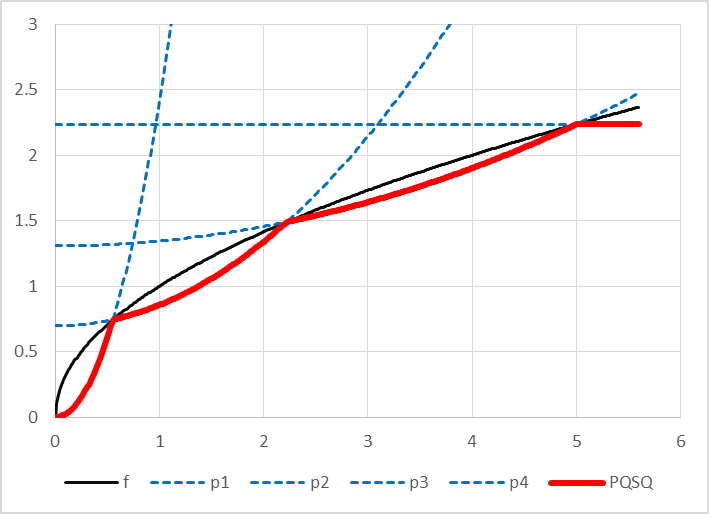
a b c

Figure . Examples of PQSQ functions with three intervals for: a , b and c. Black thin solid line is majorant function, blue dashed lines are fragment of parabolas which corresponds to different intervals, and red bold solid line is PQSQ function.

# PQSQ regression regularisation

PQSQ regularised regression coefficients can be found as minimum of function (8). Since all summands in function (8) are quadratic the solution can be found as solution of SLAE. Indeed, let us calculate derivatives of with respect to intercept:

|  |  |
| --- | --- |
|  | (14) |

After some algebra we can write equation (5). If we consider centralized data:

|  |  |
| --- | --- |
|  | (15) |

where is matrix of original non-centralised data and is vector of non-centralised response variable then we have and equation (5) can be rewritten as

|  |  |
| --- | --- |
|  | (16) |

To find intercept for regression of non-centralised data it is necessary to take

|  |  |
| --- | --- |
|  | (17) |

Further we consider the problem for centralised data. Let us find derivative of with respect to regression coefficients:

|  |  |
| --- | --- |
|  | (18) |

where is defined as , is one of the boundaries defined for th PQSQ function . After some algebra we can rewrite (18) as

|  |  |
| --- | --- |
|  | (19) |

SLAE (19) can be rewritten in matrix form:

|  |  |
| --- | --- |
|  | (20) |

where is diagonal matrix with elements of diagonal

|  |  |
| --- | --- |
|  | (21) |

# Black hole

The main properties of lasso regression: “Because of nature of this constraint (formula (7) with ) it tends to produce some coefficients that are exactly zero…” [3]. PQSQ regression regularisation does not has this property. To provide it we apply ‘black hole’ approach: if then we remove attribute (set coefficient . Threshold can be selected by different way. We apply several approaches:

1. .
2. .
3. is user defined value.
4. .

# Algorithm for PQSQ regression regularisation

Algorithm is loop of iterative solution of SLAE (21):

1. Set element of matrix of indices Q to -1 (impossible values).
2. Solve SLAE (6) (SLAE (21) with zero matrix ).
3. Identify new set of indices Q
4. While new Q is not the same as old Q do
   1. Form matrix
   2. Solve SLAE (21)
   3. Identify new set of indices Q

Condition to stop iteration is complete coincides of index matrices for two consequent iterations.

# Comparison of lasso and PQSQ regularized regression

To compare lasso, elastic net and PQSQ regularized regression we use two databases: Prostate Cancer (PC) [6] and Breast Cancer Wisconsin (BCW) [7, 8]. We use standard MatLab function lasso as implementation of lasso and elastic map. For better comparison we use PQSQ regularized regression with set of calculated by lasso function.

For PC comparison of time is presented in Table 1. Results are calculated by script TimeTestPClasso.m:

%Time Test for lasso and PQSQRegularRegr

%Use 100 of starts for each method to provide better accuracy of

%measurements.

tic;

for k=1:100

[B,FitInfo] = lasso(Prostata(:,1:8),Prostata(:,9));

end

toc/100

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostata(:,1:8),Prostata(:,9),...

'Lambda',FitInfo.Lambda);

end

toc/100

tic;

for k=1:100

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Prostata(:,1:8),Prostata(:,9),...

'Lambda' ,'Regular','lasso1');

end

toc/100

Table 1. Comparison of time of execution for lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| lasso | 0.0873 | 100% | 1.00 |
| PQSQRegularRegr(…,'lasso') | 0.0272 | 31% | 3.21 |
| PQSQRegularRegr(…,'lasso1') | 0.0348 | 40% | 2.51 |

To compare sparsity of methods we use script SparsityTest.m:

%Test standard lasso

[B,FitInfo] = lasso(Prostata(:,1:8),Prostata(:,9));

%Test PQSQRegularRegr lasso without trimming

[PqB,PqFitInfo] = PQSQRegularRegr(Prostata(:,1:8),Prostata(:,9));

%Test PQSQRegularRegr lasso with possible trimming

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Prostata(:,1:8),Prostata(:,9),...

'Regular','lasso1');

%Define array for results

sparsity = zeros(9,3);

for k=1:9 %k is number of nonzero coefficients plus 1

%Standard lasso

sparsity(k,1) = bestError( FitInfo, k-1 );

sparsity(k,2) = bestError( PqFitInfo, k-1 );

sparsity(k,3) = bestError( Pq1FitInfo, k-1 );

end

function bestError is

function err = bestError( FitInfo, nonZero )

%bestError serches the minimal value of FitInfo.MSE for elements which

%corresponds to FitInfo.DF==nonZero (with the same number of nonzero

%coefficients).

ind = FitInfo.DF == nonZero;

if any(ind)

%There is at leas one lambda with nonZero nonzero coefficients

err = min(FitInfo.MSE(ind));

else

%Absolutely impossible value as a sign of absence

err = -1;

end

end

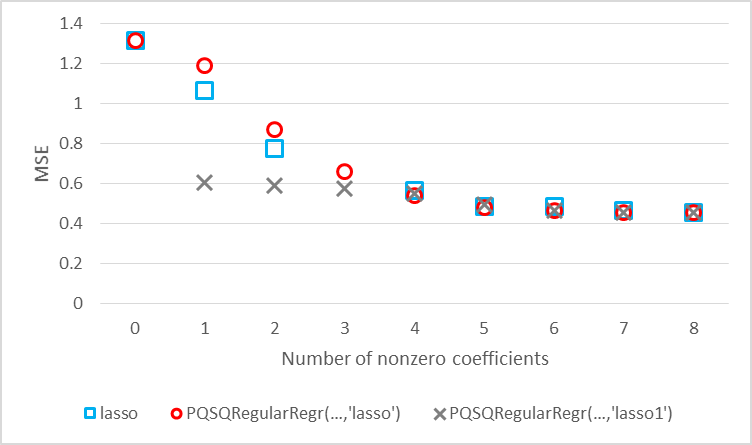


Figure . Comparison of sparsity of lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

Figure 2 shows that PQSQ imitation of lasso with trimming provide approximately the same or less Mean Square Error (MSE) for most number of nonzero coefficients but require 2.5 times less time for calculation.

For BCW comparison of time is presented in Table 2. Results are calculated by script TimeTestBCWlasso.m:

%Time Test for lasso and PQSQRegularRegr

%Use 100 of starts for each method to provide better accuracy of

%measurements. For standard lasso we use 1 start because it is slow enough

tic;

for k=1:1

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1));

end

toc/1

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda);

end

toc/100

tic;

for k=1:100

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular','lasso1');

end

toc/100

Table 2. Comparison of time of execution for lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| lasso | 29.6877 | 100.00% | 1.00 |
| PQSQRegularRegr(…,'lasso') | 0.0657 | 0.22% | 451.87 |
| PQSQRegularRegr(…,'lasso1') | 0.0694 | 0.23% | 427.78 |

To compare sparsity of methods we use script SparsityTestBCW.m:

%Test standard lasso

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1));

%Test PQSQRegularRegr lasso without trimming

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1));

%Test PQSQRegularRegr lasso with possible trimming

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Regular','lasso1');

%Define array for results

sparsity = zeros(9,3);

for k=1:9 %k is number of nonzero coefficients plus 1

%Standard lasso

sparsity(k,1) = bestError( FitInfo, k-1 );

sparsity(k,2) = bestError( PqFitInfo, k-1 );

sparsity(k,3) = bestError( Pq1FitInfo, k-1 );

end

Results of sparsity comparison are presented in Figure 3.

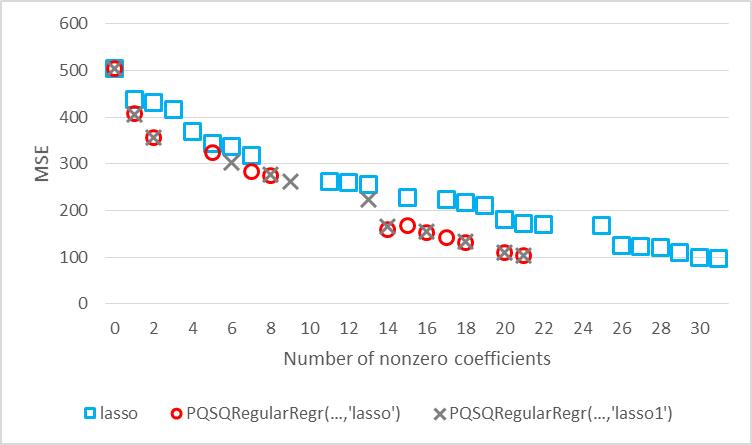


Figure . Comparison of sparsity of lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

Figure 3 shows that both PQSQ regressions are at least not worse than lasso but default value of threshold To check the results with changing of we apply test by scripts TimeTestBCWlassoEps.m:

%Time Test for lasso and PQSQRegularRegr

%Use 100 of starts for each method to provide better accuracy of

%measurements. For standard lasso we use 1 start because it is slow enough

tic;

for k=1:1

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1));

end

toc/1

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Epsilon',Inf);

end

toc/100

tic;

for k=1:100

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular','lasso1','Epsilon',Inf);

end

toc/100

and SparsityTestBCWEps.m:

%Test standard lasso

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1));

%Test PQSQRegularRegr lasso without trimming

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Epsilon',Inf);

%Test PQSQRegularRegr lasso with possible trimming

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Regular','lasso1','Epsilon',Inf);

%Define array for results

sparsity = zeros(9,3);

for k=1:32 %k is number of nonzero coefficients plus 1

%Standard lasso

sparsity(k,1) = bestError( FitInfo, k-1 );

sparsity(k,2) = bestError( PqFitInfo, k-1 );

sparsity(k,3) = bestError( Pq1FitInfo, k-1 );

end

Results of these tests are presented in Table 3 and Figure 4.

Table 3. Comparison of time of execution for lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| lasso | 29.8326 | 100.49% | 1.00 |
| PQSQRegularRegr(…,'lasso') | 0.0545 | 0.18% | 544.73 |
| PQSQRegularRegr(…,'lasso1') | 0.0622 | 0.21% | 479.62 |

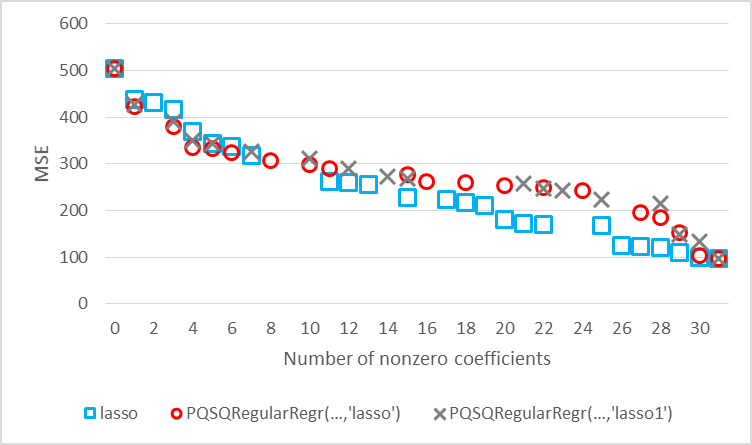


Figure . Comparison of sparsity of lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

We can see that for version with black hole with dependency from PQSQRegularRegr has less accuracy for small . If we compare Figure 4 with Figure 3 we can note that for case with fixed accuracy of PQSQRegularRegr better than for standard lasso but PQSQRegularRegr cannot form regression with more than 21 nonzero coefficient. Statistics of standard linear regression coefficients for BCW which is calculated by command

mdl = fitlm(Breast(:,2:end),Breast(:,1));

shows (see Table 4) that all coefficients which are zero for any for PQSQRegularRegr (these values are highlighted by yellow background) have *p*-values at least 63%. It means that for all these coefficients differences from zero are not statistically significant.

Table 4. Statistics of standard linear regression coefficients for BCW

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute | Coefficient | Standard error | T Statistics | *p*-value | PQSQ coefficient |
| x1 | 66.339 | 58.1370 | 1.14110 | 0.2717 | 71.081 |
| x2 | -0.318 | 3.7116 | -0.08565 | 0.9329 | 0 |
| x3 | -9.217 | 8.2904 | -1.11170 | 0.2838 | -9.353 |
| x4 | 0.085 | 0.2197 | 0.38597 | 0.7049 | 0.051 |
| x5 | 2719.400 | 1375.9000 | 1.97640 | 0.0668 | 2566.476 |
| x6 | 390.430 | 546.3800 | 0.71457 | 0.4859 | 320.136 |
| x7 | 1101.300 | 594.7500 | 1.85170 | 0.0839 | 996.805 |
| x8 | -2266.400 | 1019.3000 | -2.22340 | 0.0420 | -2047.547 |
| x9 | 593.250 | 367.4900 | 1.61430 | 0.1273 | 537.259 |
| x10 | -7941.500 | 3477.0000 | -2.28400 | 0.0374 | -7557.191 |
| x11 | -292.190 | 180.8200 | -1.61590 | 0.1269 | -279.948 |
| x12 | -11.501 | 23.6980 | -0.48530 | 0.6345 | 0 |
| x13 | 85.681 | 30.8460 | 2.77770 | 0.0141 | 83.390 |
| x14 | -1.545 | 0.7167 | -2.15590 | 0.0477 | -1.495 |
| x15 | -6104.700 | 5415.5000 | -1.12730 | 0.2773 | -5374.429 |
| x16 | -1609.800 | 1383.6000 | -1.16350 | 0.2628 | -1406.815 |
| x17 | -300.220 | 1735.1000 | -0.17303 | 0.8649 | 0 |
| x18 | 818.990 | 5005.2000 | 0.16363 | 0.8722 | 0 |
| x19 | 433.150 | 1594.0000 | 0.27173 | 0.7895 | 0 |
| x20 | 3365.700 | 12119.0000 | 0.27773 | 0.7850 | 0 |
| x21 | 2.525 | 26.9790 | 0.09359 | 0.9267 | 0 |
| x22 | 0.974 | 2.8729 | 0.33915 | 0.7392 | 0 |
| x23 | -5.001 | 2.8595 | -1.74890 | 0.1007 | -4.757 |
| x24 | 0.153 | 0.1606 | 0.95278 | 0.3558 | 0.158 |
| x25 | 688.550 | 694.4400 | 0.99152 | 0.3372 | 603.837 |
| x26 | 82.417 | 185.9600 | 0.44320 | 0.6640 | 71.124 |
| x27 | -157.840 | 181.1000 | -0.87157 | 0.3972 | -171.021 |
| x28 | -80.461 | 441.9400 | -0.18206 | 0.8580 | 0 |
| x29 | -245.340 | 262.4900 | -0.93468 | 0.3648 | -200.121 |
| x30 | 1501.100 | 1796.9000 | 0.83535 | 0.4166 | 1799.807 |
| x31 | -0.187 | 2.1897 | -0.08537 | 0.9331 | 0 |

# Comparison of elastic net and PQSQ regularized regression

We use standard MatLab function lasso as implementation of elastic map. For better comparison we use PQSQ regularized regression with set of calculated by lasso function. We compare results of calculation for parameter of mixing of lasso and ridge summands (see (7)).

For PC comparison of time is presented in Table 5. Results are calculated by script TimeTestPCelast.m:

%Time Test for elastic net and PQSQRegularRegr

%Use 100 of starts for each method to provide better accuracy of

%measurements.

tic;

for k=1:100

[B,FitInfo] = lasso(Prostate(:,1:8),Prostate(:,9),'Alpha',0.5);

end

toc/100

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet',0.5});

end

toc/100

tic;

for k=1:100

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet3',0.5});

end

toc/100

Table 5. Comparison of time of execution for lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| elastic net | 0.0875 | 100% | 1.00 |
| PQSQRegularRegr(…,'elasticnet') | 0.0405 | 46% | 2.16 |
| PQSQRegularRegr(…,'elasticnet3') | 0.0548 | 63% | 1.60 |

To compare sparsity of methods we use script SparsityTestEN.m:

%Test standard lasso

[B,FitInfo] = lasso(Prostate(:,1:8),Prostate(:,9),'Alpha',0.5);

%Test PQSQRegularRegr elastic net without trimming

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Regular',{'elasticnet',0.5});

%Test PQSQRegularRegr elastic net with possible trimming

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Regular',{'elasticnet3',0.5});

%Define array for results

sparsity = zeros(9,3);

for k=1:9 %k is number of nonzero coefficients plus 1

%Standard lasso

sparsity(k,1) = bestError( FitInfo, k-1 );

sparsity(k,2) = bestError( PqFitInfo, k-1 );

sparsity(k,3) = bestError( Pq1FitInfo, k-1 );

end

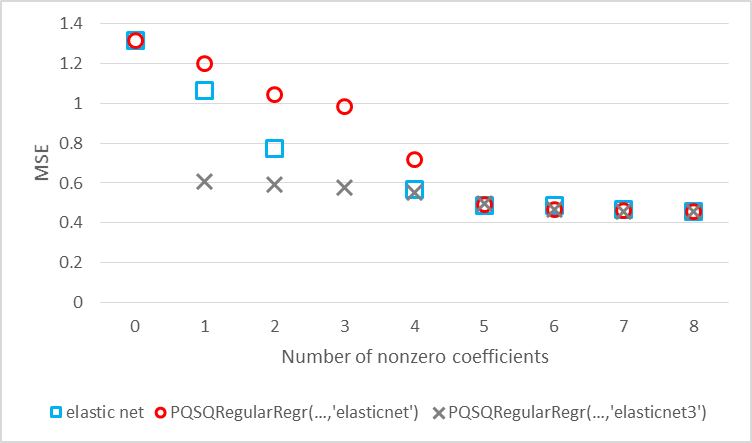


Figure . Comparison of sparsity of lasso, PQSQRegularRegr(…,'elasticnet') and PQSQRegularRegr(…,''elasticnet'3')

Figure 5 shows that PQSQ imitation of elastic net with trimming provide approximately the same or less Mean Square Error (MSE) for most number of nonzero coefficients but require 1.6 times less time for calculation.

For BCW comparison of time is presented in Table 6. Results are calculated by script TimeTestBCWelast.m:

%Time Test for elastic net and PQSQRegularRegr

%Use 100 of starts for each method to provide better accuracy of

%measurements.

tic;

for k=1:1

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1),'Alpha',0.5);

end

toc/1

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet',0.5});

end

toc/100

tic;

for k=1:100

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet3',0.5});

end

toc/100

Table 6. Comparison of time of execution for lasso, PQSQRegularRegr(…,'lasso') and PQSQRegularRegr(…,'lasso1')

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| lasso | 9.5606 | 32.20% | 3.11 |
| PQSQRegularRegr(…,'elasticnet') | 0.0715 | 0.24% | 415.21 |
| PQSQRegularRegr(…,'elasticnet3') | 0.0817 | 0.28% | 363.37 |

To compare sparsity of methods we use script SparsityTestBCWEN.m:

%Test standard lasso

[B,FitInfo] = lasso(Breast(:,2:end),Breast(:,1),'Alpha',0.5);

%Test PQSQRegularRegr lasso without trimming

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet',0.5});

%Test PQSQRegularRegr lasso with possible trimming

[Pq1B,Pq1FitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',FitInfo.Lambda,'Regular',{'elasticnet3',0.5});

%Define array for results

sparsity = zeros(9,3);

for k=1:32 %k is number of nonzero coefficients plus 1

%Standard lasso

sparsity(k,1) = bestError( FitInfo, k-1 );

sparsity(k,2) = bestError( PqFitInfo, k-1 );

sparsity(k,3) = bestError( Pq1FitInfo, k-1 );

end

Results of sparsity comparison are presented in Figure 6.

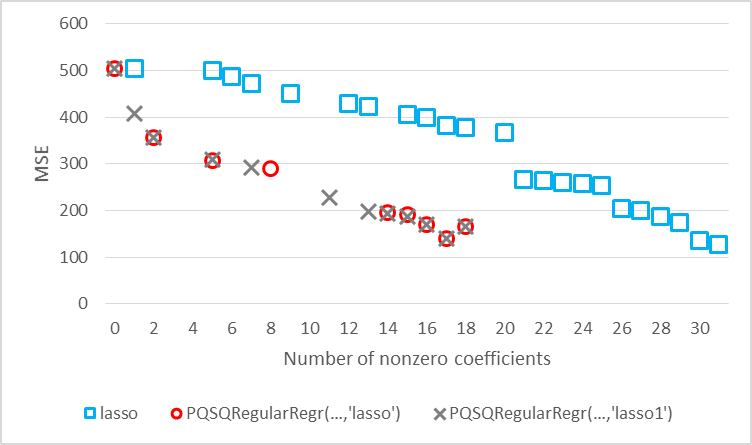


Figure . Comparison of sparsity of lasso, PQSQRegularRegr(…,'elasticnet') and PQSQRegularRegr(…,''elasticnet'3')

Figure 6 shows that both PQSQ regressions are at least not worse than lasso but cannot form model with more than 18 nonzero coefficients. All coefficients of standard linear regression which correspond to zero coefficients of PQSQ regression are statistically insignificant.

# Comparison of PQRQ regularized regressions with different regularizations

Just for playing I compare several PQSQ regressions:

1. Pseudo lasso: PQSQRegularRegr(...,'Regular','lasso1');
2. Pseudo elastic net: PQSQRegularRegr(...,'Regular',{'elasticnet3',0.5});
3. Pseudo ridge: PQSQRegularRegr(...,'Regular',{1,@L2,5,1});
4. : PQSQRegularRegr(...,'Regular',{1,@L1\_5,5,1});
5. Logarithmic: PQSQRegularRegr(...,'Regular',{1,@LLog,5,1});
6. Square root: PQSQRegularRegr(...,'Regular',{1,@LSqrt,5,1});
7. All together: PQSQRegularRegr(..., 'Regular', {'elasticnet3',0.5}, 'Regular', {0.5,@L1\_5,5,1}, 'Regular', {0.5,@LLog,5,1}, 'Regular', {0.5,@LSqrt,5,1});

To compare time of calculation for PC we implement script TimeTestPQSQPC.m:

%Time Test of different PQSQ regularized regressions

%Pseudo lasso

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular','lasso1');

end

toc/100

%pseudo elastic net with alpha = 0.5

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5});

end

toc/100

%pseudo ridge

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@L2,5,1});

end

toc/100

%Unnamed 3/2

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@L1\_5,5,1});

end

toc/100

%Unnamed log

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@LLog,5,1});

end

toc/100

%Unnamed sqrt

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@LSqrt,5,1});

end

toc/100

%All together

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5},...

'Regular',{0.5,@L1\_5,5,1},'Regular',{0.5,@LLog,5,1},...

'Regular',{0.5,@LSqrt,5,1});

end

toc/100

Results of script work are presented in Table 7.

Table . Comparison of calculation times for different regularization

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| Pseudo lasso | 0.0360 | 100.00% | 1.00 |
| Pseudo elastic net | 0.0548 | 152.22% | 0.66 |
| Pseudo ridge | 0.0302 | 83.89% | 1.19 |
|  | 0.0343 | 95.28% | 1.05 |
| Logarithmic | 0.0352 | 97.78% | 1.02 |
| Square root | 0.0371 | 103.06% | 0.97 |
| All together | 0.1112 | 308.89% | 0.32 |

For BCW time comparison implemented in script TimeTestPQSQBCW.m:

%Time Test of different PQSQ regularized regressions

%Pseudo lasso

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular','lasso1');

end

toc/100

%pseudo elastic net with alpha = 0.5

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5});

end

toc/100

%pseudo ridge

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@L2,5,1});

end

toc/100

%Unnamed 3/2

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@L1\_5,5,1});

end

toc/100

%Unnamed log

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@LLog,5,1});

end

toc/100

%Unnamed sqrt

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@LSqrt,5,1});

end

toc/100

%All together

tic;

for k=1:100

[PqB,PqFitInfo] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5},...

'Regular',{0.5,@L1\_5,5,1},'Regular',{0.5,@LLog,5,1},...

'Regular',{0.5,@LSqrt,5,1});

end

toc/100

Results of script work are presented in Table 8.

Table 8. Comparison of calculation times for different regularization

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Time (s) | Fraction of lasso (%) | Times faster |
| Pseudo lasso | 0.0674 | 100.00% | 1.00 |
| Pseudo elastic net | 0.0825 | 122.40% | 0.82 |
| Pseudo ridge | 0.0550 | 81.60% | 1.23 |
|  | 0.0628 | 93.18% | 1.07 |
| Logarithmic | 0.0731 | 108.46% | 0.92 |
| Square root | 0.0778 | 115.43% | 0.87 |
| All together | 0.1774 | 263.20% | 0.38 |

Sparsity comparison is performed by script SparsityTestPQSQPC.m:

[B1,FitInfo1] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular','lasso1');

[B2,FitInfo2] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5});

[B3,FitInfo3] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@L2,5,1});

[B4,FitInfo4] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@L1\_5,5,1});

[B5,FitInfo5] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@LLog,5,1});

[B6,FitInfo6] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{1,@LSqrt,5,1});

[B7,FitInfo7] = PQSQRegularRegr(Prostate(:,1:8),Prostate(:,9),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5},...

'Regular',{0.5,@L1\_5,5,1},'Regular',{0.5,@LLog,5,1},...

'Regular',{0.5,@LSqrt,5,1});

sparsity = zeros(9,7);

for k=1:9 %k is number of nonzero coefficients plus 1

sparsity(k,1) = bestError( FitInfo1, k-1 );

sparsity(k,2) = bestError( FitInfo2, k-1 );

sparsity(k,3) = bestError( FitInfo3, k-1 );

sparsity(k,4) = bestError( FitInfo4, k-1 );

sparsity(k,5) = bestError( FitInfo5, k-1 );

sparsity(k,6) = bestError( FitInfo6, k-1 );

sparsity(k,7) = bestError( FitInfo7, k-1 );

end

Results of sparsity comparison for PC are presented in Figure 7.

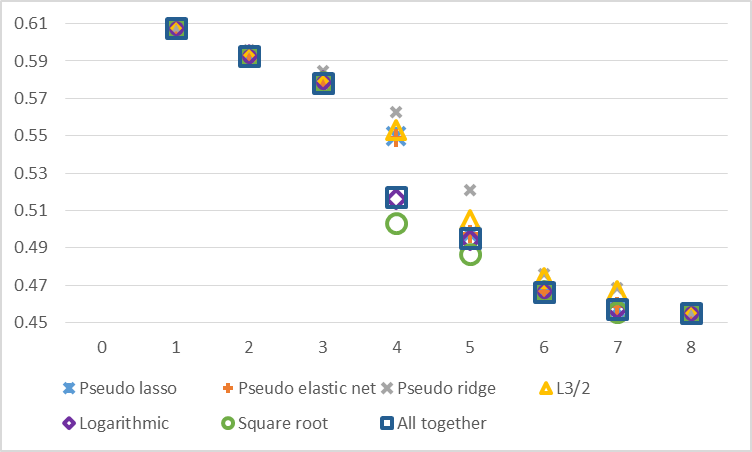


Figure 7. Comparison of sparsity for PC and different regularisations

Sparsity comparison is performed by script SparsityTestPQSQBCW.m:

[B1,FitInfo1] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular','lasso1');

[B2,FitInfo2] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5});

[B3,FitInfo3] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@L2,5,1});

[B4,FitInfo4] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@L1\_5,5,1});

[B5,FitInfo5] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@LLog,5,1});

[B6,FitInfo6] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{1,@LSqrt,5,1});

[B7,FitInfo7] = PQSQRegularRegr(Breast(:,2:end),Breast(:,1),...

'Lambda',lambda,'Regular',{'elasticnet3',0.5},...

'Regular',{0.5,@L1\_5,5,1},'Regular',{0.5,@LLog,5,1},...

'Regular',{0.5,@LSqrt,5,1});

sparsity = zeros(32,7);

for k=1:32 %k is number of nonzero coefficients plus 1

sparsity(k,1) = bestError( FitInfo1, k-1 );

sparsity(k,2) = bestError( FitInfo2, k-1 );

sparsity(k,3) = bestError( FitInfo3, k-1 );

sparsity(k,4) = bestError( FitInfo4, k-1 );

sparsity(k,5) = bestError( FitInfo5, k-1 );

sparsity(k,6) = bestError( FitInfo6, k-1 );

sparsity(k,7) = bestError( FitInfo7, k-1 );

end

Results of sparsity comparison for PC are presented in Figure 8.

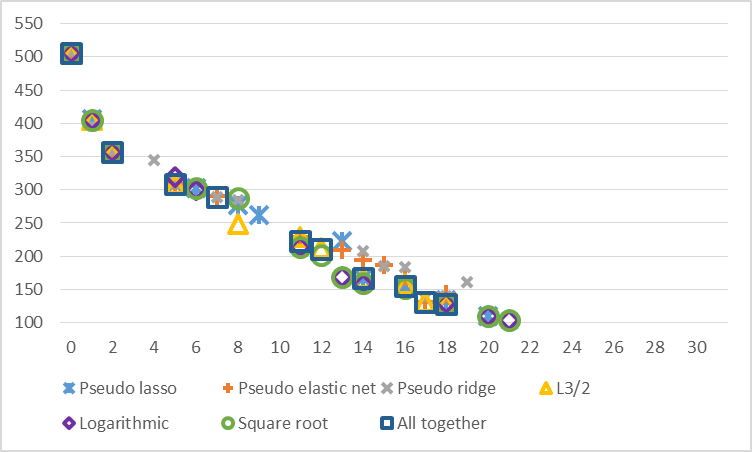


Figure 8. Comparison of sparsity for PC and different regularisations

# Test of MatLab lasso function

Let us check solution which is calculated by MatLab lasso function. To simplicity we take Prostate Cancer database and renormalize it by subtructind mean and division on square root of sum of squared centralized values (ProstateNormalization.m):

%Normalisation of prostate cancer database

%Centralization

ProstateNorm = bsxfun(@minus, Prostate, mean(Prostate));

%Rescaling

ProstateNorm = bsxfun(@rdivide, ProstateNorm, sqrt(sum(ProstateNorm.^2)/size(Prostate,1)));

This preliminary normalization allows us direct comparison of regression coefficients and Mean Squared Error (MSE). We will use function TotalError(coef, X, lambda) which calculate

|  |  |
| --- | --- |
|  | (22) |

which corresponds to lasso function of interest for centralized data (for centralized data intercept is equal to zero). Function TotalError.m is

function err = TotalError(coef, X, Y, lambda)

%TotalError calculates lasso function of interest for centralized data

%matrix X and response vector Y, vector of regression coefficients coef and

%weight of lasso regularization term lambda.

% X is data matrix with rows correspond to data points.

% Y is column vector of responses

% coef is column vector of regression coefficients

%MSE calculation

err = sum((Y - X\*coef).^2)/size(Y,1);

%Add lasso term

err = err + lambda\*sum(abs(coef));

end

Vectorised version of this function is

function err = TotalErrors(coef, X, Y, lambda)

%TotalError calculates lasso function of interest for centralized data

%matrix X and response vector Y, vector of regression coefficients coef and

%weight of lasso regularization term lambda.

% X is data matrix with rows correspond to data points.

% Y is column vector of responses

% coef is matrix of column vectors of regression coefficients

%MSE calculation

err = sum((bsxfun(@minus,X\*coef,Y)).^2)/size(Y,1);

%Add lasso term

err = err + lambda\*sum(abs(coef));

end

The next step is calculation of solution by standard lasso function and by our lassoPQSQ and PQSQRegularRegr with the same set of lambdas (script TestLassoAndPQSQ.m):

%Test of lasso

%Create array for results

res = zeros(100,4);

%Calculate solution by lasso and fix lambdas

[B,FitInfo] = lasso(ProstateNorm(:,1:8),ProstateNorm(:,9));

lambda = FitInfo.Lambda;

%Lasso uses ascending order of lambda and lassoPQSQ and PQSQRegularRegr

%use ascending one. To uniformity it is necessary to flip lasso results.

res(:,1) = flip(lambda');

res(:,2) = flip(TotalErrors(B,ProstateNorm(:,1:8),ProstateNorm(:,9)...

,lambda)');

lambda = flip(lambda);

[B3,FitInfo3] = lassoPQSQ(ProstateNorm(:,1:8),ProstateNorm(:,9),...

'Lambda',lambda,'trimming',1);

res(:,3) = TotalErrors(B3,ProstateNorm(:,1:8),ProstateNorm(:,9),lambda)';

[B4,FitInfo4] = PQSQRegularRegr(ProstateNorm(:,1:8),ProstateNorm(:,9),...

'Lambda',lambda);

res(:,4) = TotalErrors(B4,ProstateNorm(:,1:8),ProstateNorm(:,9),lambda)';

Results of calculation are presented in Table 7. We can see that lasso never over perform PQSQRegularRegr and for most of lambda is worse than lassoPQSQ. Difference between lassoPQSQ and PQSQRegularRegr is in different number of intervals: lassoPQSQ uses 4 intervals before trimming and PQSQRegularRegr uses 5 intervals before trimming. As a result threshold to set coefficient to zero for PQSQRegularRegr is approximately twice less.

Table 8. Results of calculation of lasso function of interest for standard MatLab implementation of lasso and lassoPQSQ and PQSQ regularized regression.

| Lambda | lasso function of interest | | | Number of nonzero coefficients | | |
| --- | --- | --- | --- | --- | --- | --- |
| lasso | lassoPQSQ | PQSQRegularRegr | lasso | lassoPQSQ | PQSQRegularRegr |
| 0.734460 | 1.0000 | 0.8970 | 0.8882 | 0 | 3 | 3 |
| 0.669213 | 0.9521 | 0.8647 | 0.8582 | 1 | 3 | 3 |
| 0.609762 | 0.9084 | 0.8335 | 0.8290 | 1 | 3 | 3 |
| 0.555592 | 0.8686 | 0.8035 | 0.8006 | 1 | 3 | 3 |
| 0.506235 | 0.8324 | 0.7747 | 0.7731 | 1 | 3 | 3 |
| 0.461262 | 0.7993 | 0.7471 | 0.7467 | 1 | 3 | 3 |
| 0.420285 | 0.7693 | 0.7209 | 0.7213 | 1 | 3 | 3 |
| 0.382948 | 0.7418 | 0.6960 | 0.6971 | 1 | 3 | 3 |
| 0.348928 | 0.7145 | 0.6807 | 0.6781 | 2 | 4 | 4 |
| 0.317930 | 0.6883 | 0.6573 | 0.6601 | 2 | 4 | 5 |
| 0.289686 | 0.6645 | 0.6353 | 0.6378 | 2 | 4 | 5 |
| 0.263951 | 0.6427 | 0.6146 | 0.6168 | 2 | 4 | 5 |
| 0.240503 | 0.6191 | 0.5952 | 0.5972 | 3 | 4 | 5 |
| 0.219137 | 0.5973 | 0.5771 | 0.5789 | 3 | 4 | 5 |
| 0.199670 | 0.5774 | 0.5602 | 0.5618 | 3 | 4 | 5 |
| 0.181931 | 0.5593 | 0.5445 | 0.5456 | 3 | 4 | 5 |
| 0.165769 | 0.5428 | 0.5299 | 0.5306 | 3 | 4 | 5 |
| 0.151043 | 0.5278 | 0.5181 | 0.5167 | 3 | 5 | 5 |
| 0.137624 | 0.5141 | 0.5051 | 0.5039 | 3 | 5 | 5 |
| 0.125398 | 0.5010 | 0.4931 | 0.4920 | 5 | 5 | 5 |
| 0.114258 | 0.4886 | 0.4820 | 0.4811 | 5 | 5 | 5 |
| 0.104108 | 0.4772 | 0.4718 | 0.4710 | 5 | 5 | 5 |
| 0.094859 | 0.4668 | 0.4624 | 0.4629 | 5 | 5 | 6 |
| 0.086432 | 0.4574 | 0.4538 | 0.4538 | 5 | 5 | 6 |
| 0.078754 | 0.4488 | 0.4462 | 0.4454 | 5 | 6 | 6 |
| 0.071758 | 0.4410 | 0.4383 | 0.4376 | 5 | 6 | 6 |
| 0.065383 | 0.4338 | 0.4310 | 0.4304 | 5 | 6 | 6 |
| 0.059574 | 0.4273 | 0.4243 | 0.4238 | 5 | 6 | 6 |
| 0.054282 | 0.4214 | 0.4181 | 0.4177 | 5 | 6 | 6 |
| 0.049460 | 0.4157 | 0.4124 | 0.4121 | 6 | 6 | 6 |
| 0.045066 | 0.4099 | 0.4072 | 0.4069 | 6 | 6 | 6 |
| 0.041062 | 0.4046 | 0.4024 | 0.4022 | 6 | 6 | 6 |
| 0.037414 | 0.3999 | 0.3980 | 0.3984 | 6 | 6 | 7 |
| 0.034091 | 0.3955 | 0.3939 | 0.3942 | 6 | 6 | 7 |
| 0.031062 | 0.3915 | 0.3902 | 0.3903 | 6 | 6 | 7 |
| 0.028303 | 0.3879 | 0.3869 | 0.3866 | 7 | 7 | 7 |
| 0.025788 | 0.3846 | 0.3835 | 0.3832 | 7 | 7 | 7 |
| 0.023497 | 0.3815 | 0.3803 | 0.3801 | 7 | 7 | 7 |
| 0.021410 | 0.3788 | 0.3774 | 0.3773 | 7 | 7 | 7 |
| 0.019508 | 0.3763 | 0.3748 | 0.3746 | 7 | 7 | 7 |
| 0.017775 | 0.3736 | 0.3723 | 0.3722 | 8 | 7 | 7 |
| 0.016196 | 0.3711 | 0.3700 | 0.3699 | 8 | 7 | 7 |
| 0.014757 | 0.3688 | 0.3679 | 0.3679 | 8 | 7 | 7 |
| 0.013446 | 0.3667 | 0.3660 | 0.3660 | 8 | 7 | 7 |
| 0.012252 | 0.3648 | 0.3643 | 0.3642 | 8 | 7 | 7 |
| 0.011163 | 0.3631 | 0.3627 | 0.3626 | 8 | 7 | 7 |
| 0.010171 | 0.3615 | 0.3612 | 0.3612 | 8 | 7 | 7 |
| 0.009268 | 0.3601 | 0.3599 | 0.3598 | 8 | 7 | 7 |
| 0.008445 | 0.3587 | 0.3586 | 0.3586 | 8 | 7 | 7 |
| 0.007694 | 0.3575 | 0.3575 | 0.3575 | 8 | 7 | 7 |
| 0.007011 | 0.3564 | 0.3564 | 0.3562 | 8 | 7 | 8 |
| 0.006388 | 0.3555 | 0.3555 | 0.3552 | 8 | 7 | 8 |
| 0.005820 | 0.3545 | 0.3546 | 0.3544 | 8 | 7 | 8 |
| 0.005303 | 0.3537 | 0.3538 | 0.3536 | 8 | 7 | 8 |
| 0.004832 | 0.3530 | 0.3531 | 0.3528 | 8 | 7 | 8 |
| 0.004403 | 0.3523 | 0.3525 | 0.3522 | 8 | 7 | 8 |
| 0.004012 | 0.3517 | 0.3518 | 0.3516 | 8 | 7 | 8 |
| 0.003655 | 0.3511 | 0.3513 | 0.3510 | 8 | 7 | 8 |
| 0.003331 | 0.3506 | 0.3508 | 0.3505 | 8 | 7 | 8 |
| 0.003035 | 0.3501 | 0.3503 | 0.3500 | 8 | 7 | 8 |
| 0.002765 | 0.3497 | 0.3499 | 0.3496 | 8 | 7 | 8 |
| 0.002520 | 0.3493 | 0.3495 | 0.3492 | 8 | 7 | 8 |
| 0.002296 | 0.3489 | 0.3492 | 0.3489 | 8 | 7 | 8 |
| 0.002092 | 0.3486 | 0.3489 | 0.3486 | 8 | 7 | 8 |
| 0.001906 | 0.3483 | 0.3486 | 0.3483 | 8 | 7 | 8 |
| 0.001737 | 0.3480 | 0.3483 | 0.3480 | 8 | 7 | 8 |
| 0.001582 | 0.3478 | 0.3481 | 0.3478 | 8 | 7 | 8 |
| 0.001442 | 0.3475 | 0.3478 | 0.3475 | 8 | 7 | 8 |
| 0.001314 | 0.3473 | 0.3476 | 0.3473 | 8 | 7 | 8 |
| 0.001197 | 0.3472 | 0.3475 | 0.3472 | 8 | 7 | 8 |
| 0.001091 | 0.3470 | 0.3473 | 0.3470 | 8 | 7 | 8 |
| 0.000994 | 0.3468 | 0.3471 | 0.3468 | 8 | 7 | 8 |
| 0.000905 | 0.3467 | 0.3470 | 0.3467 | 8 | 7 | 8 |
| 0.000825 | 0.3466 | 0.3469 | 0.3466 | 8 | 7 | 8 |
| 0.000752 | 0.3464 | 0.3468 | 0.3464 | 8 | 7 | 8 |
| 0.000685 | 0.3463 | 0.3467 | 0.3463 | 8 | 7 | 8 |
| 0.000624 | 0.3462 | 0.3466 | 0.3462 | 8 | 7 | 8 |
| 0.000569 | 0.3462 | 0.3465 | 0.3462 | 8 | 7 | 8 |
| 0.000518 | 0.3461 | 0.3464 | 0.3461 | 8 | 7 | 8 |
| 0.000472 | 0.3460 | 0.3463 | 0.3460 | 8 | 7 | 8 |
| 0.000430 | 0.3459 | 0.3462 | 0.3459 | 8 | 7 | 8 |
| 0.000392 | 0.3459 | 0.3462 | 0.3459 | 8 | 7 | 8 |
| 0.000357 | 0.3458 | 0.3461 | 0.3458 | 8 | 7 | 8 |
| 0.000325 | 0.3458 | 0.3461 | 0.3458 | 8 | 7 | 8 |
| 0.000297 | 0.3457 | 0.3460 | 0.3457 | 8 | 7 | 8 |
| 0.000270 | 0.3457 | 0.3460 | 0.3457 | 8 | 7 | 8 |
| 0.000246 | 0.3456 | 0.3460 | 0.3456 | 8 | 7 | 8 |
| 0.000224 | 0.3456 | 0.3459 | 0.3456 | 8 | 7 | 8 |
| 0.000204 | 0.3456 | 0.3459 | 0.3456 | 8 | 7 | 8 |
| 0.000186 | 0.3455 | 0.3459 | 0.3455 | 8 | 7 | 8 |
| 0.000170 | 0.3455 | 0.3458 | 0.3455 | 8 | 7 | 8 |
| 0.000155 | 0.3455 | 0.3458 | 0.3455 | 8 | 7 | 8 |
| 0.000141 | 0.3455 | 0.3458 | 0.3455 | 8 | 7 | 8 |
| 0.000128 | 0.3455 | 0.3458 | 0.3455 | 8 | 7 | 8 |
| 0.000117 | 0.3454 | 0.3458 | 0.3454 | 8 | 7 | 8 |
| 0.000107 | 0.3454 | 0.3457 | 0.3454 | 8 | 7 | 8 |
| 0.000097 | 0.3454 | 0.3457 | 0.3454 | 8 | 7 | 8 |
| 0.000088 | 0.3454 | 0.3457 | 0.3454 | 8 | 7 | 8 |
| 0.000081 | 0.3454 | 0.3457 | 0.3454 | 8 | 7 | 8 |
| 0.000073 | 0.3454 | 0.3457 | 0.3454 | 8 | 7 | 8 |

# MatLab implementation

MatLab implementation of PQSQ regularized regression contains one main function PQSQRegularRegr and three service functions L1, L1\_5, and L2.

Service functions calculate value of majorant function for specified :

L1.m is linear function

L1\_5.m is norm: .

L2.m is norm: .

Main function PQSQRegularRegr has following syntax:

B = PQSQRegularRegr(X, Y)

B = PQSQRegularRegr(X, Y, Name, Value)

[B, FitInfo] = PQSQRegularRegr(X, Y)

[B, FitInfo] = PQSQRegularRegr(X, Y, Name, Value)

Inputs

X is numeric matrix with n rows and p columns. Each row represents one observation, and each column represents one predictor (variable).

Y is numeric vector of length n, where n is the number of rows of X. Y(i) is the response to row i of X.

Name, Value is one or more pairs of name and value. There are several possible names with corresponding values:

'Lambda' is vector of Lambda values. It will be returned in return argument FitInfo in descending order. The default is to have PQSQRegularRegr generate a sequence of lambda values, based on 'NumLambda' and 'LambdaRatio'. PQSQRegularRegr will generate a sequence, based on the values in X and Y, such that the largest LAMBDA value is just sufficient to produce all zero coefficients B in standard lasso. You may supply a vector of real, non-negative values of lambda for PQSQRegularRegr to use, in place of its default sequence. If you supply a value for 'Lambda', 'NumLambda' and 'LambdaRatio' are ignored.

'NumLambda' is the number of lambda values to use, if the parameter 'Lambda' is not supplied (default 100). It is ignored if 'Lambda' is supplied. PQSQRegularRegr may return fewer fits than specified by 'NumLambda' if the residual error of the fits drops below a threshold percentage of the variance of Y.

'LambdaRatio' is ratio between the minimum value and maximum value of lambda to generate, if the parameter "Lambda" is not supplied. Legal range is [0,1). Default is 0.0001. If 'LambdaRatio' is zero, PQSQRegularRegr will generate its default sequence of lambda values but replace the smallest value in this sequence with the value zero. 'LambdaRatio' is ignored if 'Lambda' is supplied.

'Standardize' is indicator whether to scale X prior to fitting the model sequence. This affects whether the regularization is applied to the coefficients on the standardized scale or the original scale. The results are always presented on the original data scale. Possible values are true (any nonzero number) and false (zero). Default is TRUE, do scale X. Note: X and Y are always centred.

'PredictorNames' is a cell array of names for the predictor variables, in the order in which they appear in X. Default: {}

'Weights' is vector of observation weights. It must be a vector of non-negative values, of the same length as columns of X. At least two values must be positive. Default (1/N)\*ones(N,1).

'Epsilon' is positive value which specify minimal nonzero value of regression coefficient. It means that attribute with absolute value of regression coefficient which is less than 'epsilon' is removed from regressions (coefficient becomes zero). There are three possible ways to spesify epsilon:

positive value means that it is 'epsilon'.

zero means that r(1)/2 is used as epsilon. r(1) is right border of the first interval.

negative value means that lambda\*r(1)/2 is used as epsilon.

Default is 0.

'Regular' is description of one regularisation term in equation (8). If this parameter is omitted then 'lasso' is used. There are several special values of this argument and general case for customization:

Cell array C is a method to specify arbitrary term. Elements of array have following meaning:

C(1) is for defined term, string 'elasticnet' or 'elasticnet1'. Let us consider numeric C(1).

C(2) is handle of majorant function, for example, L1 or L2. If this argument is omitted then L1 function is used.

C(3) is array of intervals boundaries or positive integer number. If C(3) is empty, then it is interpreted as 5. If C(3) is array of intervals boundaries, then it is row vector The first element must be zero. All other elements must be sorted in ascending order. If C(3) is array of intervals then element C(4) is ignored. If C(3) is positive number then it is interpreted as number of intervals p. Let us M is specified trimming threshold. In this case intervals boundaries are calculated by following rule: the first element is zero, all other borders are calculated as r(i) = M\*(i-1)^2/p^2. Default value is 5.

C(4) is interpreted if C(3) is positive integer number only. In this case C(4) must be positive real number which interpreted as multiplier for trimming threshold definition: trimming threshold M is product of delta and maximal value of regression coefficients for OLS method (5) and (6). Default value is 1.

{'elasticnet', num} is imitation of elastic net with parameter equals num and without trimming. Num must be real number between zero and one exclusively. It is equivalent to consequence of two arrays: {num, @L1, 5, 2} and {1-num, @L2, 1, 2}.

{'elasticnet1', num} is imitation of elastic net with parameter equals num and without trimming on ridge term and with possible trimming in lasso term. Num must be real number between zero and one exclusively. It is equivalent to consequence of two arrays: {num, @L1, 5, 1} and {1-num, @L2, 1, 2}.

{'elasticnet2', num} is imitation of elastic net with parameter equals num and without trimming on lasso term and with possible trimming in ridge term. Num must be real number between zero and one exclusively. It is equivalent to consequence of two arrays: {num, @L1, 5, 2} and {1-num, @L2, 1, 1}.

{'elasticnet3', num} is imitation of elastic net with parameter equals num and with possible trimming. Num must be real number between zero and one exclusively. It is equivalent to consequence of two arrays: {num, @L1, 5, 1} and {1-num, @L2, 1, 1}.

'lasso' is simplest imitation of lasso without trimming. It is equivalent to array {1, @L1, 5, 2} or {1, @L1, [], 2}.

'lasso1' is simplest imitation of lasso with possible trimming. It is equivalent to array {1, @L1, 5, 1} or {1, @L1}.

'ridge' is simplest imitation of ridge regression without trimming. It is equivalent to array {1, @L2, 1, 2}.

'ridge1' is simplest imitation of ridge regression with possible trimming. It is equivalent to array {1, @L2, 1, 1}.

Return values:

B is the fitted coefficients for each model. B will have dimension PxL, where P = size(X,2) is the number of predictors, and L = length(lambda).

FitInfo is a structure that contains information about the sequence of model fits corresponding to the columns of B. STATS contains the following fields:

'Intercept' is the intercept term for each model. Dimension is 1xL.

'Lambda' is the sequence of lambda penalties used, in ascending order. Dimension is 1xL.

'Regularization' is cell array of cell arrays for each regularization in form {C1, C2, C3} where C1 is normalized weight of regularization term, C2 function handle for majorant function and C3 is set of intervals.

'DF' is the number of nonzero coefficients in B for each value of lambda. Dimension is 1xL.

'MSE' is the mean squared error of the fitted model for each value of lambda. Otherwise, 'MSE' is the mean sum of squared residuals obtained from the model with B and FitInfo.Intercept.

'PredictorNames' is a cell array of names for the predictor variables, in the order in which they appear in X.

'PredictorNames' is array of values epsilon which is used for each lambda.

# Reference

1. Linear regression ???
2. [Tibshirani, R. (1996). Regression shrinkage and selection via the lasso.](http://www-stat.stanford.edu/~tibs/lasso/lasso.pdf) J. Royal. Statist. Soc B., Vol. 58, No. 1, pages 267-288).
3. Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67, no. 2 (2005): 301-320.
4. Efron, B., Johnstone, I., Hastie, T. and Tibshirani, R. (2002). Least angle regression pdf file. Published in Annals of Statistics 2003
5. PQSQ???
6. Stamey, Thomas A., John N. Kabalin, John E. McNeal, Iain M. Johnstone, Fuad Freiha, Elise A. Redwine, and Norman Yang. "Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate II radical prostatectomy treated patients." *The Journal of urology* 141, no. 5 (1989): 1076-1083.
7. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science
8. W. N. Street, O. L. Mangasarian, and W.H. Wolberg. An inductive learning approach to prognostic prediction. In A. Prieditis and S. Russell, editors, Proceedings of the Twelfth International Conference on Machine Learning, pages 522--530, San Francisco, 1995. Morgan Kaufmann.