Using of Ellipsoid Method and Linear Regression with L$_1$-Regularization for Medical Data Investigation

Petro Stetsyuk[1], Viktor Stovba[1], Ivan Senko[1], and Illya Chaikovsky[1]

[1] V.M. Glushkov Institute of Cybernetics of the NASU, Academician Glushkov Avenue, 40, Kyiv, 03187, Ukraine

Abstract

The problem of finding of parameters of linear regression model with L_1-regularization and the least moduli criterion with $1 \leq p \leq 2$ is considered. To solve the problem the Shor's ellipsoid method is used, which is implemented as the emlmpr algorithm. A series of three computational experiments is conducted, which demonstrate solving time of the emlmpr algorithm and robustness of the least moduli criterion if p is close to 1. The third experiment considers situation when the model contains linearly dependent features and shows the effect of L_1-regularization on the quality of solutions obtained.

Keywords

linear regression, least moduli criterion, L_1-regularization, non-smooth optimization problem, Shor's ellipsoid method, dependent factors, data prediction

1. Introduction

Regression models are an extremely prevalent tool for effective prediction both in machine learning and artificial intelligence in general. Applying of linear regression models for building effective forecasting models, which describe linear relationships between factors, in such fields as statistics, medicine, economics, ecology, identification of parameters of complex systems etc. is studied and investigated. This type of models has proven themselves to be flexible in construction and to provide clear interpretation of relationships between dependent variable and model factors, sometimes even outperforming more complex nonlinear models [1].

When working with regression models, it is rather important to choose correct criteria for estimating model parameters. The most well-known and common variants are the criterion based on least squares and based on least moduli. Effectiveness of the first variant is confirmed by theoretical studies [2] and numerous statistical experiments. Nevertheless, one of the most significant disadvantages of the least squares criterion is the increase of the effect of large errors when they are squared, which makes the model extremely sensitive to anomalous observations (or outliers). An important condition for using this criterion is the standard normal distribution of model errors, which is not always fulfilled in practice. A well-known and effective alternative to this criterion is the criterion based on the least moduli, which is robust to outliers [3, 4] and assumes a Laplacian distribution of model errors.

Another important aspect of work with linear regression models is the presence of dependencies between two or more factors of a model, which negatively affect the quality of the obtained parameter estimates. Usually, such dependencies are detected at the stage of data preprocessing and model building by selecting optimal set of model factors that best describe relationship between the dependent variable and the factors. However, in practice, situations often occur when a certain group of factors collectively affects the dependent variable. As a result, both the criterion based on the least squares and the least moduli incorrectly determines parameters of the model, often significantly

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2024), September 25–27, 2024, Cambridge, MA, USA

stetsyukp@gmail.com (P. Stetsyuk); vik.stovba@gmail.com (V. Stovba); statistic.roots.2013@gmail.com (I. Senko); illya.chaikovsky@gmail.com (I. Chaikovsky)

0000-0003-4036-2543 (P. Stetsyuk); 0000-0003-3023-5815 (V. Stovba); 0000-0002-2432-4582 (I. Senko); 0000-0002-4152-0331 (I. Chaikovsky)

CEUR Workshop Proceedings (CEUR-WS.org)

overestimating or underestimating them. Therefore, it is expedient to develop methods and criteria that make it possible to detect such dependencies between factors and make their coefficients be close to zero. One of the most famous so-called shrinkage methods in machine learning [1] is regularization approach that permits to balance the model and reduce the effect of dependent factors on the quality of parameter determination.

The article is dedicated to applying of the Shor's ellipsoid method for finding parameters of a linear regression model with L_1-regularization and the least moduli criterion with $1 \leq p \leq 2$. This criterion includes the use of the least moduli ($p = 1$) and the least squares ($p = 2$) criteria, as well as allows to use any value of the parameter p. Certain work results of applying the ellipsoid method for this type of problems are given in [5].

2. Finding of linear regression model parameters using the least moduli criterion powered to *p*

Let us consider a classical linear regression problem: to find n unknown parameters $x_1, \dots, x_n$ with known observations $(\mathbf{a}_i, y_i)$, $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{in}) \in \boldsymbol{R}^n$, $y_i \in \boldsymbol{R}$, $i = \overline{1, m}$, which are related as follows:

$$y_i = \sum\nolimits_{j=1}^{n} a_{ij} x_j + \varepsilon_i, \;\; i = \overline{1, m}, \tag{1}$$

where a_{ij} are known coefficients, ε_i are unknown random variables, which have (approximately) the same distribution functions, $m > n$. The equation (1) can be rewritten in matrix form

$$y = Ax + \varepsilon, \tag{2}$$

where $y = (y_1, \dots, y_m)^T \in \boldsymbol{R}^m$ and $\varepsilon = (\varepsilon_1, \dots, \varepsilon_m)^T \in \boldsymbol{R}^m$ are m-dimentional vectors, A is a $m \times n$-matrix, $x = (x_1, \dots, x_n)^T \in \boldsymbol{R}^n$ is a n-dimentional vector that is to be evaluated.

The least moduli method powered to p, which corresponds to finding the unknown vector x_p^* according to the least moduli criterion powered to p $(1 \leq p \leq 2)$, is a mathematical programming problem:

$$f^* = f(x_p^*) = \min_{x \in \boldsymbol{R}^n} \left\{ f(x) = \sum\nolimits_{i=1}^{m} \left| y_i - \sum\nolimits_{j=1}^{n} a_{ij} x_j \right|^p \right\}, \tag{3}$$

where $|\cdot|$ is an absolute value of a number. The function $f(x)$ is non-smooth, if $p = 1$ and smooth, if $p > 1$.

The problem (3) is a problem of unconditional minimization of the convex function $f(x)$, subgradient of which at the point $\bar{x}$ is calculated using the following formula:

$$g_f(\bar{x}) = \left\{ \begin{matrix} p \sum\nolimits_{i=1}^{m} sign\left(\sum\nolimits_{j=1}^{n} a_{ij} \bar{x}_j - y_i \right) \left| \sum\nolimits_{j=1}^{n} a_{ij} \bar{x}_j - y_i \right|^{p-1} a_{i1}, \\ \dots \\ p \sum\nolimits_{i=1}^{m} sign\left(\sum\nolimits_{j=1}^{n} a_{ij} \bar{x}_j - y_i \right) \left| \sum\nolimits_{j=1}^{n} a_{ij} \bar{x}_j - y_i \right|^{p-1} a_{in}, \end{matrix} \right\} \tag{4}$$

If $p = 1$, the problem (3) can be formulated as a following mathematical programming problem:

$$f_1^* = \min_{x \in \boldsymbol{R}^n} \left\{ f_1(x) = \sum\nolimits_{i=1}^{m} \left| y_i - \sum\nolimits_{j=1}^{n} a_{ij} x_j \right| \right\}. \tag{5}$$

The problem (5) is a problem of unconditional minimization of convex piecewise-linear function $f_1(x)$, which corresponds to the least moduli method, which has proven to be robust to anomalous observations or outliers [3, 6]. Finding the best according to the least moduli criterion vector x^*, where x^* is a solution of the problem (5), can be formulated as the following LP-problem: to find

$$f_1^* = \min_{z \in \boldsymbol{R}^n,\, z \geq 0} \sum\nolimits_{i=1}^{m} z_i$$

$$\text{subject to} \quad y_i - \sum\nolimits_{j=1}^{n} a_{ij} x_j \leq z_i, \; -y_i + \sum\nolimits_{j=1}^{n} a_{ij} x_j \leq z_i, \; i = \overline{1,m}. \tag{6}$$

For solving the LP-problem (6) one can use appropriate standard linear programming tools. At the same time, as we find the vector x^* we find optimal values of the vector $z^* = (z_1^*, \dots, z_m^*)^T$ as well, elements of which define estimates for independent random variable ε_i, $i = \overline{1,m}$.

If $p = 2$ the problem (3) can be written as the following mathematical programming problem:

$$f_2^* = \min_{x \in \boldsymbol{R}^n} \left\{ f_2(x) = \sum\nolimits_{i=1}^{m} \left(y_i - \sum\nolimits_{j=1}^{n} a_{ij} x_j \right)^2 \right\}. \tag{7}$$

The problem (7) is a problem of unconditional minimization of a convex quadratic function $f_2(x)$, which corresponds to the least squares method. Linear independency of the rows of the matrix A provides existence of an analytical solution $x^* = (A^T A)^{-1} A^T y$ of the problem (7). Otherwise, if rows of the matrix A are linearly dependent or $n > m$, it is impossible to obtain an analytical solution. In that case one can use methods for balancing the model, in particular, regularization.

Let us consider the problem (3) with L_1-regularization:

$$f_p^* = f_p(x_p^*) = \min_{x \in \boldsymbol{R}^n} \left\{ f_p(x) = \sum\nolimits_{i=1}^{m} \left| y_i - \sum\nolimits_{j=1}^{n} a_{ij} x_j \right|^p + \lambda \sum\nolimits_{j=1}^{n} |x_j| \right\}. \tag{8}$$

The problem (8) is a problem of unconditional minimization of a convex piecewise-linear function $f_p(x)$. Here λ is a regularization parameter, and if $\lambda = 0$ the function $f_p(x)$ coincides with the function $f(x)$. To calculate the subgradient of the function $f_p(x)$ at the point $\bar{x}$ one can use the following formula:

$$g_{f_p}(\bar{x}) = g_f(\bar{x}) + \lambda\, sign(\bar{x}), \tag{9}$$

where $g_f(\bar{x})$ is calculated using the expression (4).

For solving the problem (8) the Shor's ellipsoid method [7, 8, 9] can be used, which is implemented as the emshor program [10]. We will apply it for the problem of the function $f_p(x)$ minimization, providing that its minimum point x_p^* is localized in n-dimensional ball with radius r_0, which is centered at the point $x_0 \in \boldsymbol{R}^n$, i.e. $\|x_0 - x_p^*\| \leq r_0$. The algorithm to be used is called emlmpr, description of which is given below.

3. The emlmpr algorithm and its Octave implementation

The input parameter of the algorithm is $\varepsilon_f > 0$ – accuracy, with which $f_p^* = f_p(x_p^*)$ is to be found.

Initialization. Let us consider $n \times n$-matrix B and set $B_0 := I_n$, where I_n is $n \times n$ identity matrix. We go to the first iteration with values x_0, r_0 and B_0. Let values $x_k \in \boldsymbol{R}^n$, r_k, B_k be found at the iteration k. Passing to the iteration $k + 1$ consists of the following sequence of actions.

Step 1. Calculate $f_p(x_k)$ and subgradient $g_{f_p}(x_k)$ at the point x_k using formula (9). If $r_k \left\| B_k^T g_{f_p}(x_k) \right\| \leq \varepsilon_f$, then "Stop: $k^* = k$ and $x_p^* = x_k$". Otherwise, go to the step 2.

Step 2. Set $\xi_k := \frac{B_k^T g_{f_p}(x_k)}{\left\| B_k^T g_{f_p}(x_k) \right\|}$.

Step 3. Calculate the next point

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{where} \quad h_k = \frac{1}{n+1} r_k.$$

Step 4. Calculate

$$B_{k+1} := B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) (B_k \xi_k) \xi_k^T \quad \text{and} \quad r_{k+1} := r_k \frac{n}{\sqrt{n^2 - 1}}.$$

Step 5. Go to the iteration $k + 1$ with values x_{k+1}, r_{k+1}, B_{k+1}.

Theorem. *Sequence of points* $\{x_k\}_{k=0}^{k^*}$ *satisfy the following inequalities:*

$$\left\| B_k^{-1}(x_k - x_p^*) \right\| \le r_k, \quad k = 0,1,2,\dots,k^*.$$

On each iteration $k > 0$ the value of decreasing of volume of the ellipsoid $E_k = \{x \in R^n : \left\| B_k^{-1}(x_k - x) \right\| \le r_k\}$, which localizes point x_p^*, is constant and equal to

$$q = \frac{vol(E_k)}{vol(E_{k-1})} = \sqrt{\frac{n-1}{n+1}} \left(\frac{n}{\sqrt{n^2-1}} \right)^n < exp\left\{ -\frac{1}{2(n+1)} \right\} < 1.$$

Theorem implies the fact that the algorithm of finding x_p^* can be successfully run on modern computers, if $n = 10 \div 30$ and $m = 100 \div 1000$. Indeed, to decrease in 10 times volume of the ellipsoid localizing the point x_p^*, it is needed to perform K iterations, where $K = \frac{ln\,10}{ln\,q} \approx (2\,ln\,10)(n+1) \approx 4.6(n+1)$. It means that in order to improve deviation of found record value of the function $f_p(x)$ from its optimal value f_p^* by 10 times, it is necessary to perform $4.6(n+1)^2$ iterations of the algorithm for finding x_p^*.

If $n = 30$ and $\varepsilon_f = 10^{-6} \times f(x_0)$, then the maximal number of iterations of the algorithm is equal to $4.6(n+1)^2 = 46 \times 961 = 44206$. Therefore, even the straight-up matrix-vector implementation of calculation of the function $f_p(x)$ value and its subgradient according to the formula (9) allows to provide fast algorithm work on modern computers.

The algorithm emlmpr for finding an approximation to the point x_p^* is implemented using Octave language. Its code is given below.

```
# Input parameters:                                               #com01
# A(m,n) - observation matrix;                                    #com02
# y(m,1) - vector of tags (output vector);                        #com03
# p - power for least moduli criterion, 1<=p<=2;                  #com04
# lambda - regularization rate;                                   #com05
# x0(n,1) - starting point;                                       #com06
# r0 - radius of the ball centered at x0 that localizes x_p^*;    #com07
# epsf, maxitn - stop parameters:                                 #com08
# epsf - precision to stop by the value of the function fp,       #com09
# maxitn - maximal number of iterations;                          #com10
# intp - print information for every intp iteration.              #com11
# Output parameters:                                              #com12
# xp(n,1) - approximation to x_p^*;                               #com13
# fp - the value of the function f_R at the point xp;             #com14
# itn - the number of iterations;                                 #com15
# ist - exit code: 1 - epsf, 4 - maxitn.                          #com16
function [xp,fp,itn,ist] = emlmpr(A,y,p,lambda,x0,r0,
                                 epsf,maxitn,intp);               #row01
   n = columns(A); xp = x0; B = eye(n); r = r0;                   #row02
   dn = double(n); beta = sqrt((dn-1.d0)/(dn+1.d0));              #row03
   for (itn = 0:maxitn)                                           #row04
     temp = A*xp-y; fp = sum(abs(temp).^p) + lambda*sum(abs(xp)); #row05
     if((mod(itn,intp)==0)&&(intp<=maxitn))                       #row06
        printf(" itn %4d  fp %14.6e\n",itn,fp);                   #row07
     endif                                                        #row08
     g1 = p*A'*(sign(temp).*(abs(temp)).^(p-1)) + lambda*sign(xp);#row09
     g = B'*g1; dg = norm(g);                                     #row10
     if(r*dg < epsf) ist = 1;  return; endif                      #row11
     xi = (1.d0/dg)*g; dx = B * xi;                               #row12
     hs = r/(dn+1.d0); xp -= hs * dx;                             #row13
     B += (beta - 1) * B * xi * xi';                              #row14
     r = r/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);                 #row15
   endfor                                                         #row16
   ist = 4;                                                       #row17
endfunction                                                       #row18
```

Core of the emlmpr program is the for loop (rows 4–16). First, the value of the function f (line 5) and its normalized subgradient at the point x_p (row 10) are calculated. If the stop condition is satisfied (row 11), the algorithm stops its work. Stop in the emlmpr algorithm occurs when a condition $r_k \left\| B_k^T g_{f_p}(x_k) \right\| \le \varepsilon_f$ is fulfilled, which is equivalent to condition $f_p(x_k) - f_p^* \le \varepsilon_f$. Otherwise, the next point x_{k+1} is calculated (row 13), the space transformation matrix B_{k+1} (row 14) and the radius r_{k+1} (row 15) are recalculated.

4. Computational experiments without regularization

To demonstrate the effectiveness of the emlmpr algorithm work we present results of three computational experiments conducted for solving the problem (8). For the first and the second experiments parameters $n = 30$ and $m = 10 \times n = 300$. The purpose of the first experiment is to estimate time of solving the problem (8) for specified parameters on a personal computer with Intel Core i7-10750H processor (2.6 GHz), and 16 Gb RAM. The purpose of the second experiment is to demonstrate robustness of the least moduli method, and therefore solutions of the problem (8) without regularization ($\lambda = 0$), if p is close to one. Third experiment is dedicated to finding parameters of linear regression model using real medical data for further prediction psychological indicators.

All the calculations are performed on a computer with Intel Core i7-10750H processor (2.6 GHz), 16 Gb RAM in Windows 10/64 system using GNU Octave, version 6.3.0. For the first two experiments regularization parameter λ is chosen equal to zero.

Test example 1. For the first experiment input data for the problem (8) are matrix A and vector y, which are generated randomly with a standard uniform distribution according to the following formulas: `A = 10*rand(m,n)`, `y = A*xstar(n,1)`, `xstar(n,1) = round(10*rand(n,1) + 0.5)`. Starting point is chosen according to the rule `x0(n,1) = round(5*rand(n,1))`, and radius of the sphere, in which the point $x_p^* = x_{star}$ is located, is chosen according to the rule `r0 = 5*norm(x0 – xstar)`, i.e. $r_0 = \|x_0 - x_{star}\|$. The first experiment is implemented by the following Octave code.

```
# Test 1: emlmpr running time for n = 30 and m = 300
n = 30, m = 10*n,
rand("seed", 2024);
A = 10*rand(m,n);

xstar = round(10.0*rand(n,1) + 0.5); y = A*xstar;
x0 = round(5.0*rand(n,1)); r0 = 5*norm(x0 - xstar),
maxitn = 50000, intp = 10000, lambda = 0.0,
# running the emlmpr algorithm for p=1.0;1.1.2;1.5;1.8;2.0
printf("\n Test 1: emlmpr runnning time for n = 30 and m = 300 \n");
epsf0 = 1.e-6; ntest = 5; table = [];
for (i = 1:ntest)
  p = 1.d0 + (i - 1.d0)/(ntest - 1.d0),
  epsf = epsf0**(p); time0 = time();
  [xp,fp,itn,ist] = emlmpr(A,y,p,lambda,x0,r0,epsf,maxitn,intp);
  time1 = time() - time0,
  dx = norm(xp - xstar);
  table = [table; p epsf time1 itn  ist fp dx];
  itn, fp,
endfor
n,m,
printf("   p      epsf    time    itn   ist       fp            dx   \n");
for (i = 1:ntest)
  printf("    %4.1f         %6.1e         %4.2f     %6d     %2d               %10.5e     %10.1e\n",
        table(i, 1:7))

endfor
```

Results of the emlmpr program work for the first experiment are *time* required to solve the problem (8) with accuracy ε_f, the number of iterations *itn* of the method, the minimum value of the function f_p found, norm of deviation dx of the found approximation to the minimum point from the known minimum point `xstar` are given in Table 1. Here ε_f is chosen as follows: if $p = 1$ the value $\varepsilon_f = 10^{-6}$, if $p > 1$ we choose $\varepsilon_f = (10^{-6})^p$.

Table 1
Results of solving the problem (8) with $n = 30$, $m = 300$ and $\lambda = 0$

p	ε_f	*time (sec)*	*itn*	f_p	dx
1.0	1.0e−06	5.17	45375	1.71062e−08	2.3e−11
1.2	3.2e−08	6.99	42148	3.58266e−10	1.2e−10
1.5	1.0e−09	5.39	40061	9.87425e−12	4.1e−10
1.8	3.2e−11	5.97	38260	1.81563e−13	7.2e−10
2.0	1.0e−12	3.91	37216	7.45098e−15	1.8e−09

It is easy to see from Table 1 that to get solution with accuracies $10^{-6} \div 10^{-12}$ for different p the emlmpr algorithm requires approximately 40 000 iterations and no more than 7 seconds of time. The least deviation dx equals 2.3e−11 and is obtained for $p = 1$.

Test example 2. The purpose of the second experiment is to demonstrate robustness of the least moduli method, which means that the same robustness will characterize solutions of the problem (8), if p is close to one. Here, the matrix A, the starting point x_0, ball radius r_0 are chosen to be the same as in the first test, the vector y is adjusted so that its odd components remain the same as in the first test, and even components are multiplied by the value `q = (1.0 + 1.0*sign(0.5 - rand))`. Thus, even components of the vector y can be considered anomalous (incorrect) results of observations.

```
# Test 2: robustness of the least moduli method for n = 30 and m = 300
n = 30, m = 10*n,
rand("seed", 2024);
A = 10*rand(m,n);
# test example generation
xstar = round(10.0*rand(n,1) + 0.5);
y = A*xstar;
x0 = round(5.0*rand(n,1)); r0 = 5*norm(x0 - xstar),
m1 = m/2,
for i = 1:m1
  ind = (i-1)*2 + 1;
  y(ind) = y(ind)*(1.0 + 1.0*sign(0.5 - rand));
endfor
# running the emlmpr algorithm for p=1.0;1.1.2;1.5;1.8;2.0
printf("\nTest 2: robustness of the Least Moduli Method \n");
maxitn = 50000, intp = 10000, lambda = 0.0,
epsf0 = 1.e-6; ntest = 5; table = [];
for (in = 1:ntest)
  p = 1.d0 + (in - 1.d0)/(ntest - 1.d0),
  epsf = epsf0**(p);
  time0 = time();
  [xp,fp,itn,ist] = emlmpr(A,y,p,lambda,x0,r0,epsf,maxitn,intp);
  time1 = time() - time0,
  dx = norm(xp - xstar);
  table = [table; p epsf time1 itn  ist fp dx fp^(1/p)];
  itn, fp,
endfor
n,m,
printf("   p      epsf   time   itn  ist      fp            dx        r(fp)\n");
for (i = 1:ntest)
  printf(" %4.1f  %6.1e  %4.2f %6d %2d   %10.5e %10.1e   %10.5e\n", table(i, 1:8))
endfor
```

Calculation results for $n = 30$ and $m = 300$ are given in Table 2. Here, ist is an exit code of the emlmpr program, dx is a norm of deviation of found approximation to the minimum point from the point `xstar`. The 5th column contains values of the function f_p at the found point x_p, the 7th column contains the p-th root of the 5th column. For all the values of the parameter p code $ist = 1$, which indicates successful completion of the program.

Table 2
Results of solving the problem (8) with $n = 30$, $m = 300$, $\lambda = 0$, and different p

p	ε_f	*time (sec)*	*itn*	f_r	dx	$\sqrt[p]{f_r}$
1.0	1.0e−06	3.60	43337	1.34006e+05	1.1e−10	1.34006e+05
1.2	3.2e−08	2.80	23909	7.26497e+05	2.8e+01	4.88638e+04
1.5	1.0e−09	3.41	28017	3.85135e+06	5.4e+01	2.45702e+04
1.8	3.2e−11	3.77	32560	2.04598e+07	6.4e+01	1.50542e+04
2.0	1.0e−12	3.03	37360	1.09408e+08	6.9e+01	1.04598e+04

Results of Table 2 show that the function value f_r grows as the parameter p increases: from 1.34e+05 if $p = 1$ to 1.09e+08 if $p = 2$. Deviation dx of the solution found from the minimum point with $p = 1$ is significantly smaller than if $p > 1$, which confirms robustness of the least moduli method corresponding to $p = 1$ situation. It is important to emphasize that this situation is typical for all the values of the parameter p close enough to 1. Time used for finding solutions for each of the parameter p values does not exceed 4 seconds.

5. Computational experiments with regularization

To show effectiveness of the emlmpr algorithm applied to real data we consider the problem of prediction of psychological indicators of the patient's condition based on cardiological data obtained using complex [11]. There were 90 patients studied with more than 200 features including cardiological and basic ones (like age and ordinal number). Willing to exclude choice of categorial features recoding method from analysis so we are omitting categorial feature as well as ordinal. Practically, usage of ordinal features instead of numerical could increase the quality of linear modelling, see [12], however, we need to simplify experiment in order to research only the ellipsoid method usage. While ability of the medical complex [11] to create binning good enough for the linear modelling is out the scope of the current research. So, we are taking just 175 numerical features that we have. Then, we apply the feature selection procedure to test the ellipsoid method on the dataset being optimal at least at some sense.

We want to select features that describe relationship between medical and psychological data in the best way using the R^2 metric [1]. While the goal of the studying the medical data includes feature interpretability, we take these data as is. In other words, we do not make transformations like PCA and similar ones to get linear independent features. Undoubtedly, it is possible to get some interpretation even after the transformations, but our approach is to take features as is. Taking into account that internal metrics for feature importance in the case of linear regression model work are the best when features are either linearly independent or have normal distributions at least, we cannot rely on internal linear regression metrics, so we try to use “wrapper” approach for the model feature selection [13]. For the quality metric, we use 5-fold cross-validation [1]. Since the initial dataset holds missing values, we use simple imputation via median strategy using only training subsample to avoid distortion due to the whole-set median calculation. Moreover, in our situation the initial number of features, which is 200, is greater than number of observations, which is 90, so we start from the first feature, increase number of features until the quality metric R^2 stops to grow. Also, we consider non-transformed features to decrease the number of experiments to perform and the variability of the whole scheme. Selection of the optimal transformation is an additional task, which is out of scope of the current paper. In general, the feature selection procedure is described at Figure 1.

The calculations for feature selection are made in Python 3 [14] using Google Colab with Sequential Feature Selection and Linear Regression classes with embedded R^2-metric taken from Scikit-learn library [15]. We also used Pandas library [16] for keeping feature names during calculations.

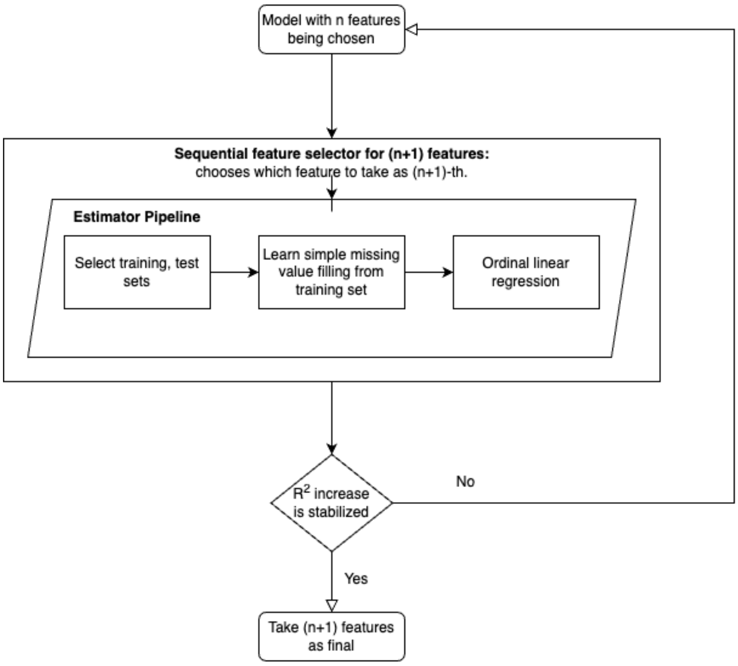
Model with n features being chosen
Sequential feature selector for (n+1) features: chooses which feature to take as (n+1)-th.
Estimator Pipeline
Select training, test sets
Learn simple missing value filling from training set
Ordinal linear regression
R² increase is stabilized
No
Yes
Take (n+1) features as final

Figure 1: Feature selection workflow

The observation matrix A consists of values of the following 16 numerical features for 90 patients: (1) observation number; (2) amplitude Q (μV) (wd. II); (3) amplitude S (μV) (wd. III); (4) amplitude P (μV) (wd. III); (5) amplitude Q (μV) (wd. AvL); (6) amplitudes R/P ratio (wd. II); (7) amplitude Q (μV) (wd. AvF); (8) LFn; (9) amplitude S (μV) (wd. AvF); (10) ECG phase ratio index; (11) state of regulation reserves; (12) withdrawal code AvR_init; (13) comprehensive assessment of occurrence of significant cardiovascular events_init; (14) functional condition according to Baevsky; (15) withdrawal code I_univ; (16) HFn; (17) target: Beck anxiety scale. The last feature is target and is to be predicted.

To determine parameters of linear regression model and further prediction the emlmpr algorithm is used with parameter $p = 1$ and $p = 2$, where the first case corresponds to the least moduli method, and the second case corresponds to the least square method. The observation matrix A is as follows:

```
A0 = [
1 0 -765 64 -120 2.04 0 46.37 -331 16 62 1 0.67 2 38 53.63 11
2 0 0 39 0 5.84 0 84.43 0 49 41 2 0.86 6 38 15.57 19
3 0 -160 26 -57 7.62 0 70.91 0 30 48 2 1.14 4 38 29.09 12
4 0 0 11 0 3.09 -67 91.19 0 30 34 2 1.14 4 38 8.81 2
5 -45 0 90 0 4.56 -55 93.7 0 13 67 1 0.4 4 38 6.3 7
6 0 0 26 0 3.23 0 85.06 -53 8 67 2 0.4 3 38 14.94 6
7 0 0 55 -179 3.51 0 68.22 0 49 78 3 2.0 3 38 31.78 6
8 0 -175 16 -60 5.46 0 82.71 0 49 48 1 0.4 5 38 17.29 2
9 0 0 21 0 4.59 0 57.3 0 35 81 3 1.56 3 38 42.7 7
10 0 0 27 0 2.67 -65 78.81 0 24 59 1 0.4 4 38 21.19 8
11 0 0 0 0 17.25 0 95.99 0 49 67 2 0.4 3 38 4.01 10
12 0 0 37 -191 7.46 0 79.07 0 49 66 1 0.4 2 38 20.93 6
13 0 0 32 -211 12.28 0 81.25 0 49 47 2 1.33 1 38 18.75 6
14 0 0 26 -200 17.15 0 71.46 0 27 74 1 0.67 3 38 28.54 2
15 0 0 18 0 7.34 0 91.2 0 49 40 2 0.67 4 38 8.8 2
16 0 -165 59 0 3.8 0 68.81 -197 49 77 1 0.4 5 38 31.19 13
17 -62 0 60 0 9.76 -74 77.9 0 49 59 3 1.6 4 38 22.1 19
18 0 0 102 0 5.9 -42 83.86 0 49 59 1 0.4 3 38 16.14 4
19 0 -135 40 -64 11.58 0 86.74 -153 49 57 2 1.0 1 38 13.26 10
20 -56 0 42 0 7.2 -54 83.35 0 29 66 1 0.67 3 38 16.65 7
21 -31 0 110 0 7.83 -36 94.87 0 49 63 1 0.4 3 38 5.13 6
22 -39 0 41 -41 8.8 0 83.32 0 49 71 1 0.4 1 38 16.68 8
23 -61 0 0 0 35.32 -70 71.37 0 0 89 1 1.14 2 38 28.63 19
24 -33 0 28 -26 3.38 0 75.29 0 12 45 2 0.86 5 38 24.71 7
25 0 0 36 0 6.78 0 90.25 0 49 74 2 1.25 3 38 9.75 3
26 0 0 30 0 7.15 0 81.1 0 20 69 1 0.4 3 38 18.9 7
27 -43 0 11 0 10.29 -34 94.77 0 0 30 1 1.0 6 38 5.23 7
28 0 0 57 0 3.05 -42 61.31 0 18 79 1 0.67 2 38 38.69 7
29 0 0 23 0 7.34 0 86.33 0 49 61 2 1.25 2 38 13.67 7
30 0 0 48 0 4.12 0 80.99 0 7 29 3 1.8 5 38 19.01 6
31 0 0 22 0 7.82 0 83.77 0 20 61 1 0.0 3 38 16.23 6
32 0 0 0 0 4.62 0 86.79 -138 16 56 2 1.33 3 38 13.21 7
33 -26 0 87 0 3.57 -25 87.02 0 34 66 2 0.86 3 38 12.98 3
34 -32 0 37 0 6.49 0 84.53 0 34 65 1 0.4 1 38 15.47 19
35 0 0 18 0 11.95 0 90.88 0 16 60 1 0.0 2 38 9.12 8
36 0 0 0 0 11.68 0 77.05 0 18 71 3 1.56 2 38 22.95 11
37 0 -564 12 -87 1.55 0 91.99 -320 3 50 1 0.4 2 38 8.01 7
38 0 -292 54 -87 4.65 0 72.96 -194 0 54 2 0.67 5 38 27.04 19
39 0 0 14 0 10.36 0 29.71 0 49 56 1 0.4 4 38 70.29 19
40 -54 -239 19 -149 6.52 0 72.46 0 49 78 1 0.4 2 38 27.54 10
41 0 0 27 -27 8.78 0 40.15 0 24 73 2 0.86 4 38 59.85 11
42 0 0 56 0 8.46 0 93.54 0 49 70 1 0.4 3 38 6.46 10
43 0 -68 34 -46 8.09 0 91.59 0 0 71 2 0.4 2 38 8.41 8
44 0 -127 40 0 5.75 0 82.6 0 18 76 2 0.67 3 38 17.4 1
45 0 -141 20 -41 4.37 0 61.08 0 16 71 2 0.86 5 38 38.92 17
46 0 0 48 0 6.98 0 94.61 0 3 40 2 1.25 5 38 5.39 19
47 0 -348 13 0 4.02 0 93.62 -127 49 70 1 0.4 0 38 6.38 19
48 0 -40 60 -36 6.31 0 54.75 0 3 61 2 1.25 6 38 45.25 3
49 0 -273 47 -74 5.47 0 88.5 -301 11 33 2 1.33 6 38 11.5 8
50 -26 0 26 0 4.62 0 82.08 0 17 67 3 1.25 3 38 17.92 8
51 0 0 26 0 13.05 0 89.35 0 25 47 2 0.86 4 38 10.65 10
52 -69 0 51 0 10.38 -71 60.73 0 49 72 2 0.86 4 38 39.27 19
53 0 0 25 -27 5.15 0 62.39 0 35 61 3 2.46 5 38 37.61 9
54 -56 0 162 0 3.9 -49 96.97 0 17 60 1 0.4 4 38 3.03 2
55 -112 0 126 0 4.9 -104 89.7 0 0 52 2 0.86 4 38 10.3 19
56 0 -472 20 -37 4.41 0 57.99 -109 23 76 1 0.0 3 38 42.01 11
57 -56 0 28 0 6.6 -34 72.68 0 33 60 1 0.4 3 38 27.32 19
58 -97 0 77 0 9.96 -107 91.48 0 0 50 3 2.31 5 38 8.52 9
59 -96 0 52 0 16.01 -77 79.61 0 49 29 2 1.25 3 38 20.39 19
60 0 0 29 0 2.59 0 96.7 0 24 42 2 1.8 6 38 3.3 8
61 0 0 14 0 6.76 0 57.36 0 49 54 2 0.67 5 38 42.64 11
62 0 0 22 0 5.84 0 61.29 257 16 76 2 0.67 2 38 38.71 6
63 0 0 58 0 8.19 0 74.21 0 12 71 2 0.67 4 38 25.79 11
64 0 0 71 0 3.8 0 73.13 0 9 66 2 0.67 4 38 26.87 4
65 0 -79 0 -32 12.91 0 89.79 0 35 73 2 0.67 2 38 10.21 17
66 -27 0 61 0 6.6 -38 83.68 0 18 50 2 1.78 3 38 16.32 3
67 0 -353 0 -32 5.87 -94 76.64 0 7 72 2 0.67 3 38 23.36 8
68 -109 0 41 -58 2.08 -75 59.33 0 34 40 1 1.14 4 38 40.67 19
69 0 0 29 0 5.6 0 91.66 0 22 51 3 1.56 4 38 8.34 13
70 -36 0 44 -31 7.62 0 68.39 0 49 81 1 0.4 3 38 31.61 11
71 0 0 36 0 6.25 0 66.75 0 1 77 1 0.4 3 38 33.25 8
72 0 0 0 0 11.3 0 93.48 0 9 50 1 1.0 5 38 6.52 6
73 0 0 0 0 12.71 -27 68.99 0 27 74 1 0.4 5 38 31.01 7
74 0 0 67 0 4.81 0 78.17 -301 13 67 1 0.4 4 38 21.83 9
75 -69 0 63 0 7.09 -85 87.27 0 49 32 1 0.4 6 38 12.73 11
76 0 0 47 0 6.78 0 95.53 0 0 43 1 0.4 5 38 4.47 6
77 0 -328 93 -34 2.46 0 79.8 -117 19 57 1 0.4 3 38 20.2 8
78 0 0 37 0 3.31 0 73.71 0 19 83 2 0.67 3 38 26.29 11
79 0 0 18 -29 6.93 0 63.65 0 25 83 2 1.25 2 38 36.35 6
80 -25 0 28 0 10.73 0 68.85 0 11 62 1 0.4 3 38 31.15 19
81 0 0 63 0 7.95 0 74.47 0 22 76 2 1.25 3 38 25.53 7
82 -128 0 61 0 9.95 -129 92.42 0 22 65 1 0.4 3 38 7.58 16
83 0 0 17 0 9.51 0 94.55 0 7 74 1 0.4 4 38 5.45 8
84 0 -308 72 -59 4.26 0 89.58 -210 49 54 1 0.4 2 38 10.42 12
85 0 0 10 0 14.3 0 88.47 0 6 49 2 0.67 4 38 11.53 7
86 0 0 43 -135 6.55 0 95.41 0 4 46 3 1.56 4 38 4.59 3
87 0 -644 0 -81 3.9 0 97.95 -442 52 41 3 2.0 4 38 2.05 17
88 0 62 21 -31 4.83 0 61.33 0 22 76 2 1.14 4 38 38.67 3
89 0 0 39 0 2.5 0 86.99 -190 49 44 1 1.14 4 38 13.01 6
90 0 0 0 0 4.68 0 77.62 0 49 76 2 0.67 2 38 22.38 10];
```

Results of the emlmpr program work is given in Table 3. It contains problem solving time (line 3), the number of iterations (line 4), value of the function at the point x_{ε}^{*} (line 5) and solution of the problem x_{ε}^{*} (line 6) for four accuracies and two values of the parameter p.

Table 3
Results of the emlmpr program work with $n = 16$, $m = 90$, $\lambda = 0$, $p = 1.0; 2.0$ and different accuracies ε_f

p	ε_f	*time (sec)*	*itn*	f_p
1.0	1.0e−06	0.56	7640	2.61732e+02
1.0	1.0e−20	0.69	8383	2.61732e+02
2.0	1.0e−12	0.95	11700	1.38880e+03
2.0	1.0e−40	2.15	29719	1.38880e+03

Table 4 shows that to solve the problem with $p = 1$ with $\varepsilon_f = 10^{-6}$ and $\varepsilon_f = 10^{-20}$ the emlmpr program requires approximately 8 thousand operations. If we use $p = 1$ for the same accuracies 11700 iterations are required, and their number is increased to 29719 iterations when using $\varepsilon_f = 10^{-40}$. The f_p value for fixed p remains unchanged.

As it can be seen from Table 4, the emlmpr program successfully finds linear regression model coefficients when using $\lambda = 0$ (see Table 5). However, some of the coefficients are rather larger than others (bold values in Table 5), which can indicate presence of dependency between the following features in the observation matrix. To reduce their effect on the quality of coefficients restoration we apply L_1-regularization, which allows to set model parameters corresponding to dependent columns to zero. In practice, it is difficult to obtain exactly zero values of the corresponding parameters, so we have to settle for values close to zero with a certain accuracy.

Table 5
Linear regression model parameters found by the emlmpr program with $\boldsymbol{p = 1.0; 2.0}$, $\boldsymbol{\lambda = 0}$ and different accuracies $\boldsymbol{\varepsilon_f}$

$p = 1$		$p = 2$	
$\varepsilon_f = 10^{-6}$	$\varepsilon_f = 10^{-20}$	$\varepsilon_f = 10^{-12}$	$\varepsilon_f = 10^{-40}$
-8.9453e-02	-8.9453e-02	-1.1440e-01	-1.1440e-01
-2.5798e-03	-2.5798e-03	-7.0401e-03	-7.0401e-03
-3.2033e-02	-3.2033e-02	-2.4929e-02	-2.4929e-02
2.0159e-02	2.0159e-02	2.7608e-02	2.7608e-02
2.4191e-01	2.4191e-01	2.6629e-01	2.6629e-01
6.8819e-03	6.8820e-03	3.5181e-02	3.5181e-02
1.4368e+08	**1.4868e+08**	**7.5756e+06**	**6.3818e+06**
-1.4482e-02	-1.4482e-02	-1.2610e-02	-1.2610e-02
3.7213e-02	3.7213e-02	3.7520e-02	3.7520e-02
-8.9244e-03	-8.9244e-03	-4.5045e-02	-4.5045e-02
1.4087e+00	1.4087e+00	2.1146e+00	2.1146e+00
-1.0175e+00	-1.0175e+00	-2.4560e+00	-2.4560e+00
-3.2982e-01	-3.2982e-01	-1.0611e-01	-1.0611e-01
-3.9263e+08	**-3.7817e+08**	**-7.8529e+06**	**-4.2703e+07**
1.4368e+08	**1.4868e+08**	**7.5756e+06**	**6.3818e+06**
5.5243e+08	**-4.9760e+08**	**-4.5914e+08**	**9.8456e+08**

Table 6 contains coefficients of linear regression model found by the emlmpr program with $p = 1.0; 2.0$, different accuracies ε_f and regularization rate $\lambda = 0.1$. Corresponding values to large coefficients from Table 5, as well as any changes in coefficients digits are highlighted in bold. It is easy to see that now these coefficients are rather close to zero with sufficient accuracy: 10^{-2} for the feature 7 with any values of p and ε_f, 10^{-8} for the feature 14 with $p = 1$ and $\varepsilon_f = 10^{-6}$ and even 10^{-29} for the feature 16 with $p = 2$ and $\varepsilon_f = 10^{-40}$. The rest of the coefficients remained almost unchanged except several digits. It is also worth noting that increasing of the regularization rate leads to decreasing coefficients values of dependent features even more. It gives an instrument to adjust the impact of regularization and obtain coefficients at dependent features close enough to zero, thus improving quality of the solutions obtained.

The prediction results obtained using the model with parameters calculated with the emlmpr algorithm show that using the least moduli method ($\boldsymbol{p = 1}$) we obtain many more zero values (which means that solution is found with required accuracy) than in case of using the least square method ($\boldsymbol{p = 2}$). Thus, using $\boldsymbol{p = 1}$ is more appropriate than $\boldsymbol{p = 2}$.

Table 6
Linear regression model parameters found by the emlmpr program with $p = 1.0; 2.0$, $\lambda = 0.1$ and different accuracies ε_f

$p = 1$		$p = 2$	
$\varepsilon_f = 10^{-6}$	$\varepsilon_f = 10^{-20}$	$\varepsilon_f = 10^{-12}$	$\varepsilon_f = 10^{-40}$
-8.9453e-02	-8.9453e-02	-1.14**38**e-01	-1.14**38**e-01
-2.5798e-03	-2.5798e-03	-7.04**85**e-03	-7.04**85**e-03
-3.2033e-02	-3.2033e-02	-2.4929e-02	-2.4929e-02
2.0159e-02	2.0159e-02	2.76**17**e-02	2.76**17**e-02
2.4191e-01	2.4191e-01	2.662**0**e-01	2.662**0**e-01
6.88**19**e-03	6.8820e-03	3.51**99**e-02	3.51**99**e-02
4.0311e-02	**4.0311e-02**	**5.6342e-02**	**5.6342e-02**
-1.4482e-02	-1.4482e-02	-1.2**595**e-02	-1.2**595**e-02
3.7213e-02	3.7213e-02	3.752**7**e-02	3.752**7**e-02
-8.924**6**e-03	-8.924**6**e-03	-4.**4948**e-02	-4.**4948**e-02
1.4087e+00	1.4087e+00	2.1**069**e+00	2.1**069**e+00
-1.0175e+00	-1.0175e+00	-2.4**451**e+00	-2.4**451**e+00
-3.2982e-01	-3.2982e-01	-1.0**552**e-01	-1.0**552**e-01
5.7126e-08	**7. 8108e-16**	**2.6027e-13**	**9.1628e-28**
1.3304e-01	**1.3304e-01**	**1.6186e-01**	**1.6186e-01**
1.4384e-08	**-2.0000e-16**	**4.6263e-14**	**-5.3458e-28**

6. Conclusions

The paper investigates the problem of finding parameters of linear regression model with the least moduli criterion with $\mathbf{1 \leq p \leq 2}$ and $\boldsymbol{L_1}$-regularization. The problem is formulated as a problem of unconditional minimization of a convex piecewise-linear function. For solving this problem, Shor's ellipsoid method is used, which is implemented by the emlmpr program using Octave programming language.

Series of three computational experiments with the emlmpr program are considered. Results of the first experiment show that the problem of finding parameters of linear regression model with $\boldsymbol{n = 30}$ and $\boldsymbol{m = 300}$ can be solved within 7 seconds being run on modern laptop of average performance. The second experiment shows that the least moduli criterion is robust if $\boldsymbol{p}$ is close to one, thus solutions of the problem are robust as well. The third experiment is dedicated to using of $\boldsymbol{L_1}$-regularization for decreasing effect of linearly dependent features that the model can include on the solutions quality. Results of the experiment, where real cardiological data are used for prediction of psychological indicators of the patient's condition, show that the emlmpr algorithm can successfully compute linear regression model parameters with $\boldsymbol{n = 16}$, $\boldsymbol{m = 90}$ within 3 seconds, and set coefficients at dependent features to zero with sufficient accuracy using $\boldsymbol{L_1}$-regularization approach.

Acknowledgements

The paper is supported by National Research Foundation of Ukraine (grants № 2021.01/0136 and №2023.04/0094), Volkswagen Foundation grant № 97775, the project of research works of young scientists №07-02/03-2023, the NASU grant for research laboratories/groups of young scientists №02/01-2024(5), and the DTT TS KNU NASU project № 0124U002162.

References

[1] G. James, D.Witten, T. Hastie, R. Tibshirani, J. Taylor, An Introduction to Statistical Learning: with Applications in Python, Springer Texts in Statistics, Springer Cham, New York, NY, 2023. doi:10.1007/978-3-031-38747-0

[2] M. Deisenroth, A. Faisal, C. Soon Ong, Mathematics for Machine Learning: textbook, Cambridge, 1st Edition, 2020.

[3] P.J. Huber, E.M. Ronchetti, Robust Statistics, John Wiley & Sons, 2nd Edition, 2011.

[4] F.H. Clarke, Optimization and Nonsmooth Analysis, SIAM, 1990.

[5] P. Stetsyuk, M. Budnyk, I. Sen'ko., V. Stovba, I. Chaikovsky, Using the Ellipsoid Method to Study Relationships in Medical Data, Cybernetics and Computer Technologies (2023) 23–43. doi:10.34229/2707-451X.23.3.3

[6] J. Fan, P. Hall, On curve estimation by minimizing mean absolute deviation and its implications, The Annals of Statistics (1994) 867–885.

[7] N.Z. Shor, Cutting-off Method with Space Dilation for Solving Convex Programming Problems, Cybernetics (1977) 94–95.

[8] N.Z. Shor, Nondifferentiable Optimization and Polynomial Problems, Kluwer, Amsterdam, 1998.

[9] N.Z. Shor, Minimization Methods for Non-Differentiable Functions, Berlin, Springer-Verlag, 1985.

[10] P. Stetsyuk, A. Fischer, O. Khomyak, The Generalized Ellipsoid Method and Its Implementation, Communications in Computer and Information Science (2020) 355–370. doi:10.1007/978-3-030-38603-0_26.

[11] I. Chaikovsky, M. Primin, A. Kazmirchuk, Development and implementation into medical practice new information technologies and metrics for analysis of small changes in electromagnetic field of human heart, Visnyk of the National Academy of Sciences of Ukraine (2021) 33–43. doi:10.15407/visn2021.02.033.

[12] R. Persson, Weight of evidence transformation in credit scoring models: How does it affect the discriminatory power? Master's thesis, Lund university, Lund, Sweden, 2021. https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9066332&fileOId=9067075

[13] L. Jundong, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, J. Tang, H. Liu, Feature Selection: A Data Perspective., ACM Computing Surveys (2017), 1–45. doi:10.1145/3136625

[14] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

[15] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12.85 (2011), 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html

[16] W. McKinney, Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, Austin, 28 June-3 July 2010, 56–61. doi: 10.25080/Majora-92bf1922-00a