

# $r$ -Алгоритм для навчання моделей лінійної регресії

Стецюк П.І.<sup>1</sup>, Вакуленко Д.В.<sup>2</sup>, Стецюк М.Г.<sup>1</sup>

<sup>1</sup>Інститут кібернетики імені В.М. Глушкова НАН України

<sup>2</sup>Тернопільський національний медичний університет  
імені І.Я. Горбачевського

Математичне та програмне  
забезпечення інтелектуальних систем  
**(МПЗІС-2023)**

22-24 листопада 2023, Дніпро

# План доповіді

- 1  $r(\alpha)$ -алгоритм та програма **ralgb5a**
- 2 Узагальнена лінійна регресія та програма **ralmpr**
- 3 Обчислювальні експерименти

# План доповіді

- 1  $r(\alpha)$ -алгоритм та програма ralg5a
- 2 Узагальнена лінійна регресія та програма ralmpr
- 3 Обчислювальні експерименти

# Що таке $r$ -алгоритми?

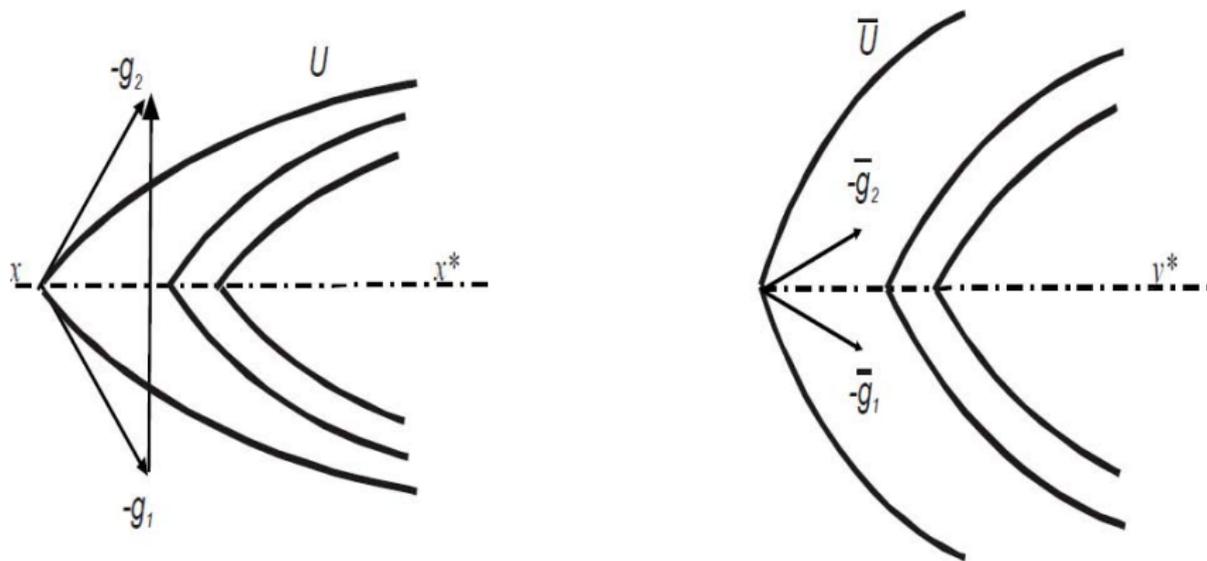
$r$ -Алгоритми призначені для знаходження наближення до точки мінімуму опуклої функції  $f(x)$ ,  $x \in R^n$ , та використовують значення  $f(x_k)$  та субградієнта  $g_f(x_k)$ .

Вони отримали таку назву від слова "різниця".

Їх називають  $r$ -алгоритмами Шора (1970), або  $r$ -алгоритмами Шора–Журбенко (1971).

-  ШОР Н.З. Методы минимизации недифференцируемых функций и их приложения. – Докт. диссерт., Киев, 1970.
-  ШОР Н.З., ЖУРБЕНКО Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов // Кибернетика. – 1971. – №3. – С. 51–59.

# Ідея $r$ -алгоритмів (за рис. із книг Шора)



Як за допомогою операції розтягу простору  
побудувати субградієнтні алгоритми – монотонні  
(майже монотонні) за функцією, що мінімізується?

Обчислювальна схема  $r(\alpha)$ -алгоритму

**Означення.**  $r(\alpha)$ -Алгоритмом називається процедура побудови послідовностей  $\{x_k\}_{k=0}^{\infty}$  та  $\{B_k\}_{k=0}^{\infty}$  за правилом:

$$x_{k+1} = x_k - h_k B_k \xi_k, \quad B_{k+1} = B_k R_{\beta}(\eta_k), \quad k = 0, 1, 2, \dots, \quad (1)$$

$$\text{де } \xi_k = \frac{B_k^T g_f(x_k)}{\|B_k^T g_f(x_k)\|}, \quad h_k \geq h_k^* = \underset{h \geq 0}{\operatorname{argmin}} f(x_k - h B_k \xi_k), \quad (2)$$

$$\eta_k = \frac{B_k^T r_k}{\|B_k^T r_k\|}, \quad r_k = g_f(x_{k+1}) - g_f(x_k), \quad \beta = \frac{1}{\alpha} < 1. \quad (3)$$

Тут  $x_0$  — стартова точка,  $B_0 = I_n$  — одинична  $n \times n$ -матриця,  $h_k$  — кроковий множник,  $\alpha > 1$  — коефіцієнт розтягу простору,  $R_{\beta}(\eta) = I_n + (\beta - 1)\eta\eta^T$  — оператор „стиснення“ простору субградієнтів в нормованому напрямку  $\eta$  з коефіцієнтом  $\beta < 1$ ,  $g_f(x_k)$ ,  $g_f(x_{k+1})$  — субградієнти функції  $f(x)$  в точках  $x_k$ ,  $x_{k+1}$ .

# Два способи регулювання кроку

1. Величина  $h_k$  вибирається з умови  $h_k = h_k^*$  ( $h_k \approx h_k^*$ ), що означає точний (наближений) одновимірний пошук мінімуму функції в напрямі спуску.

2. Величина  $h_k$  налаштовується (адаптується) в процесі виконання одновимірного пошуку мінімуму функції.

# $r(\alpha)$ -алгоритм с адаптивним кроком

виявився ефективним варіантом  $r$ -алгоритмів і на його основі розроблено ряд програмних реалізацій (Журбенко М.Г., Кунцевич О.В., Лиховид О.П. та ін.)

Для  $r(\alpha)$ -алгоритму з адаптивним кроком розроблено Octave-функції: ralb5 (2011, Стецюк П.І.), ralb4 (2016), **ralb5a** (2017, грудень 2020) – спрощена версія ralb5.



СТЕЦЮК П.И. Теория и программные реализации  $r$ -алгоритмов Шора // Кибернетика и системный анализ. – 2017. – № 5. – С. 43–57.

## Код Octave-функції ralgb5a (початок)

```

#
#   Octave-function ralgb5a (Petro Stetsyuk, 02 December 2020) #rowc01
# Input parameters: #rowc02
#   calcfg -- name of the function calcfg(x) for calculation #rowc03
#             of f(x) and its subgradient g(x), #rowc04
#   x -- starting point (it is modified in the program), x(1:n) #rowc05
#   alpha -- coefficient of space dilation (alpha = 2:4) #rowc06
#   h0, q1 -- parameters of the adaptive step adjustment, #rowc07
#   recommend: h0=1, q1=1.0 - nonsmooth, q1=0.8:0.95 - smooth #rowc08
#   epsx, epsg, maxitn -- stop parameters #rowc09
#   intp -- print information for every intp iteration #rowc10
# Output parameters: #rowc11
#   xr -- record point (with the best function value), xr(1:n) #rowc12
#   fr -- the value of the function f at the point xr #rowc13
#   itn -- the number of iterations #rowc14
#   nfg -- the number of function calcfg calls #rowc15
#   ist -- exit code: 2-epsg, 3-epsx, 4-maxitn, 5-warning #rowc16
# For more details see: Stetsyuk, P.I. Theory and Software #rowc17
# Implementations of Shor's r-Algorithms. Cybernetics and #rowc18
# Systems Analysis 53, 692-703 (2017) #rowc19
#
function [xr,fr,itn,nfg,ist] = ralgb5a(calcfg,x,alpha,h0,q1, #row001
                                     epsg,epsx,maxitn,intp); #row002
itn = 0; B = eye(length(x)); hs = h0; lsa = 0; lsm = 0; #row003
xr = x; [fr,g0] = calcfg(xr); nfg = 1; #row004
if (intp>0) #row005
    printf("itn %4d f%16.6e fr%16.6e nfg %4d\n",itn,fr,fr,nfg); #row006
endif #row007
if (norm(g0) < epsg) ist = 2; return; endif #row008

```

## Код octave-функції ralg5a (продовження)

```

for (itn = 1:maxitn)                                     #row009
  dx = B * (g1 = B' * g0)/norm(g1);                     #row010
  d = 1; ls = 0; ddx = 0;                               #row011
  while (d > 0)                                         #row012
    x -= hs * dx; ddx += hs * norm(dx);                #row013
    [f, g1] = calcfg(x); nfg ++;                       #row014
    if (f < fr) fr = f; xr = x; endif                  #row015
    if(norm(g1) < epsg) ist = 2; return; endif         #row016
    ls ++; (mod(ls,3) == 0) && (hs *= 1.1);            #row017
    if(ls > 500) ist = 5; return; endif               #row018
    d = dx' * g1;                                       #row019
  endwhile                                           #row020
  (ls == 1) && (hs *= q1); lsa=lsa+ls; lsm=max(lsm,ls); #row021
  if(mod(itn,intp)==0)                                #row022
    if (intp>0)                                        #row023
      printf("itn %4d f %14.6e fr %14.6e", itn, f, fr); #row024
      printf(" nfg %4d lsa %3d lsm %3d\n", nfg, lsa, lsm); #row025
    endif                                             #row026
    lsa=0; lsm=0;                                     #row027
  endif                                              #row028
  if(ddx < epsx) ist = 3; return; endif              #row029
  xi = (dg = B' * (g1 - g0) )/norm(dg);              #row030
  B += (1 / alpha - 1) * B * xi * xi';              #row031
  g0 = g1;                                           #row032
endfor                                              #row033
ist = 4;                                           #row034
endfunction                                         #row035

```

# Вибір параметрів для ralg5a

Для мінімізації негладких функцій рекомендується:

$$\alpha = 2 \div 4, \quad h_0 = 1.0, \quad q_1 = 1.0.$$

Якщо відома оцінка відстані від стартової точки  $x_0$  до точки мінімуму  $x^*$ , то початковий крок  $h_0$  доцільно вибирати згідно умови  $h_0 \approx \|x_0 - x^*\|$ .

Для мінімізації гладких функцій рекомендується

$$q_1 = 0.8 \div 0.95.$$

За такого вибору параметрів, як правило, середня кількість кроків за напрямком рідко перевищує два, а за  $n$  ітерацій точність за функцією покращується в три-п'ять разів.

# План доповіді

- 1  $r(\alpha)$ -алгоритм та програма `ralgb5a`
- 2 Узагальнена лінійна регресія та програма `ralmpr`
- 3 Обчислювальні експерименти

# Модель лінійної регресії ( $m > n$ )

Нехай для оцінки  $n$  невідомих параметрів  $x_1, \dots, x_n$  використовується  $m$  спостережень  $y_1, \dots, y_m$ , таких що

$$y_i = \sum_{j=1}^n a_{ij}x_j + u_i, \quad i = 1, \dots, m, \quad (4)$$

де  $a_{ij}$  – відомі коефіцієнти,  $u_i$  – невідомі випадкові величини, що мають (приблизно) однакові функції розподілу.

Рівняння (4) можна записати в матричній формі:

$$y = Ax + u, \quad (4a)$$

де  $y = (y_1, \dots, y_m)^T \in \mathbb{R}^m$ ,  $u = (u_1, \dots, u_m)^T \in \mathbb{R}^m$ ,  
 $A$  –  $m \times n$ -матриця,  $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$  – вектор параметрів, які потрібно оцінити.

# Метод найменших модулів в степені $p$

Задача знаходження невідомого вектора  $x_p^*$  за критерієм найменших модулів в степені  $p$ , де  $1 \leq p \leq 2$ , має вигляд:

$$f_{LMP}(x_p^*) = \min_{x \in \mathbb{R}^n} \left\{ f_{LMP}(x) = \sum_{i=1}^m \left| y_i - \sum_{j=1}^n a_{ij} x_j \right|^p \right\}, \quad (5)$$

де функція  $f_{LMP}(x)$  є негладкою, якщо  $p = 1$ , та  $f_{LMP}(x)$  є гладкою, якщо  $p > 1$ .

Частинними випадками задачі (5) є відомі методи:

Метод Найменших Модулів (МНМ), якщо  $p=1$ ,

Метод Найменших Квадратів (МНК), якщо  $p=2$ .

# Оптимізаційні задачі для МНМ та МНК

Для МНМ маємо задачу лінійного програмування: знайти

$$f_{LMM}^* = \min_{x \in \mathbb{R}^n, z \geq 0} \sum_{i=1}^m z_i \quad (6)$$

при обмеженнях

$$y_i - \sum_{j=1}^n a_{ij}x_j \leq z_i, \quad -y_i + \sum_{j=1}^n a_{ij}x_j \leq z_i, \quad i = 1, \dots, m. \quad (7)$$

Для МНК маємо безумовну задачу мінімізації: знайти

$$f_{LSM}^* = \min_{x \in \mathbb{R}^n} \left\{ \sum_{i=1}^m \left( \sum_{j=1}^n a_{ij}x_j - y_i \right)^2 = \|Ax - y\|^2 \right\}. \quad (8)$$

Якщо ранг матриці  $A$  дорівнює  $n$ , то  $x^* = (A^T A)^{-1} A y$ .

Узагальнена лінійна регресія ( $1 \leq p_1, p_2 \leq 2$ )

Якщо ранг матриці  $A$  менший ніж  $n$ , то задачу (5) можна замінити задачею безумовної мінімізації опуклої функції:

$$f_p(x_p^*) = \min_{x \in \mathbb{R}^n} \left\{ f_p(x) = \sum_{i=1}^m \left| y_i - \sum_{j=1}^n a_{ij} x_j \right|^{p_1} + \lambda \sum_{i=1}^n |x_i|^{p_2} \right\}, \quad (9)$$

де  $\lambda \geq 0$  – скалярний параметр регуляризації.

Функція  $f_p(x)$  є негладкою, якщо  $p_1 = 1$  або  $p_2 = 1$ , та  $f_p(x)$  є гладкою, якщо  $p_1 > 1$  та  $p_2 > 1$ .

## Lasso- і ridge- регресії та задача (9)

Частинними випадками задачі (9) є відомі задачі:

мінімізація негладкої функції (lasso-регресія):

$$f_1^* = f_1(x^*) = \min_{x \in \mathbb{R}^n} \left\{ f_1(x) = \|y - Ax\|^2 + \lambda \sum_{i=1}^n |x_i| \right\}, \quad (10)$$

мінімізація гладкої функції (ridge-регресія):

$$f_2^* = f_2(x^*) = \min_{x \in \mathbb{R}^n} \left\{ f_2(x) = \|y - Ax\|^2 + \lambda \sum_{i=1}^n x_i^2 \right\}. \quad (11)$$

Формула для субградієнта  $f_p(x)$  в задачі (9)

$$g_{f_p}(\bar{x}) = p_1 \begin{pmatrix} \sum_{i=1}^m \operatorname{sign} \left( \sum_{j=1}^n a_{ij} \bar{x}_j - y_i \right) \left| \sum_{j=1}^n a_{ij} \bar{x}_j - y_i \right|^{p_1-1} a_{i1} \\ \dots \\ \sum_{i=1}^m \operatorname{sign} \left( \sum_{j=1}^n a_{ij} \bar{x}_j - y_i \right) \left| \sum_{j=1}^n a_{ij} \bar{x}_j - y_i \right|^{p_1-1} a_{in} \end{pmatrix} +$$

$$+ \lambda p_2 \begin{pmatrix} \sum_{i=1}^m \operatorname{sign}(\bar{x}_1) |\bar{x}_1|^{p_2-1} \\ \dots \\ \sum_{i=1}^m \operatorname{sign}(\bar{x}_n) |\bar{x}_n|^{p_2-1} \end{pmatrix}.$$

# Octave-функція ralmpr

Алгоритм знаходження наближення до точки  $x_p^*$  в задачі (9) реалізовано octave програмою **ralmpr** (r-algorithm for least moduli to the power of **p** with regularization), код якої є нижче.

```
# Вхідні параметри:
# A, y, p1, lambda, p2 - дані про задачу (9), 1 <= p1, p2 <= 2
# x0(n,1), alpha, h0, q1 - стартова точка та параметри r-алгоритму
# epsx, epsg, maxitn, intr - параметри зупинки та інтервал друку
# Вихідні параметри:
# xp(n,1), fp - рекордна точка та значення функції в ній
# itn, nfg - кількості ітерацій та обчислень функції і субградієнта
# ist - код завершення: 2-epsg, 3-epsx, 4-maxitn, 5-warning
function [xp,fp,itn,nfg,ist] = ralmpr(A,y,p1,lambda,p2,x0,alpha, #row01
                                     h0,q1,epsg,epsx,maxitn,intp);
itn = 0; n=columns(A); B = eye(n); hs = h0; lsa = 0; lsm = 0; #row02
xr = x = x0; temp = A*x0 - y; fr = fp = sum(abs(temp).^p1); #row03
g0 = p1*A'*(sign(temp).*(abs(temp)).^(p1-1)); nfg = 1; #row04
fr = fp = fp + lambda*sum(abs(x).^p2); #row05
g0 = g0 + lambda*p2*(sign(x).*(abs(x)).^(p2-1)); #row06
if (intp>0) #row07
    printf("itn %4d f%16.6e fr%16.6e nfg %4d\n",itn,fr,fr,nfg); #row08
endif #row09
if (norm(g0) < epsg) ist = 2; return; endif #row10
```

## Код octave-функції ralmpr (продовження)

```

for (itn = 1:maxitn)
  dx = B * (g1 = B' * g0)/norm(g1);
  d = 1; ls = 0; ddx = 0;
  while (d > 0)
    x -= hs * dx; ddx += hs * norm(dx);
    temp = A*x - y; f = sum(abs(temp).^p1);
    g1 = p1*A'*(sign(temp).*(abs(temp)).^(p1-1));
    f = f + lambda*sum(abs(x).^p2);
    g1 = g1 + lambda*p2*(sign(x).*(abs(x)).^(p2-1));
    nfg ++;
    if (f < fr) fr = fp = f; xr = xp = x; endif
    if(norm(g1) < epsg) ist = 2; return; endif
    ls ++; (mod(ls,3) == 0) && (hs *= 1.1);
    if(ls > 500) ist = 5; return; endif
    d = dx' * g1;
  endwhile
  (ls == 1) && (hs *= q1); lsa=lsa+ls; lsm=max(lsm,ls);
  if(mod(itn,intp)==0)
    if (intp>0)
      printf("itn %4d f %14.6e fr %14.6e", itn, f, fr);
      printf(" nfg %4d lsa %3d lsm %3d\n", nfg, lsa, lsm);
    endif
    lsa=0; lsm=0;
  endif
  if(ddx < epsx) ist = 3; return; endif
  xi = (dg = B' * (g1 - g0) )/norm(dg);
  B += (1 / alpha - 1) * B * xi * xi';
  g0 = g1;
endfor
ist = 4;
endfunction

```

```

#row11
#row12
#row12
#row13
#row14
#row15
#row16
#row17
#row18
#row19
#row20
#row21
#row22
#row23
#row24
#row25
#row26
#row27
#row28
#row29
#row30
#row31
#row32
#row33
#row34
#row35
#row36
#row37
#row38
#row39
#row40

```

# План доповіді

- 1  $r(\alpha)$ -алгоритм та програма `ralgb5a`
- 2 Узагальнена лінійна регресія та програма `ralmpr`
- 3 Обчислювальні експерименти

# Про дві тестові задачі

Обчислювальні експерименти проводилися для двох тестових задач (9). Для першої задачі кількість невідомих параметрів  $n = 200$  та кількість спостережень  $m = 1000$ , а для другої задачі кількість невідомих параметрів  $n = 1000$  та кількість спостережень  $m = 200$ .

Мета обох експериментів – оцінити час розв'язання задачі (9) для вказаних параметрів на персональному комп'ютері з процесором Intel Core i5-9400f, 2.9 GHz, 16GB RAM, GNU Octave версія 5.1.0.

# Про вхідні дані тестових задач

Для першого та другого експериментів матриця  $A$  та вектор  $y$  – вхідні дані для задачі (9) генерувалися випадковим чином з рівномірним розподілом  $U(0, 1)$  за формулами:  $A = 10 * \text{rand}(m, n)$ ,  $y = A * x_{\text{star}}(n, 1)$ ,  
 $x_{\text{star}}(n, 1) = \text{round}(10 * \text{rand}(n, 1) + 0.5)$ ;  
стартова точка  $x_0(n, 1) = \text{round}(5 * \text{rand}(n, 1))$ ;  
параметри  $r$ -алгоритму:  
 $\alpha = 2.0$ ,  $h_0 = 1.0$ ,  $q_1 = 1.0$ ,  $\text{epsg} = 1.d-16$ ,  
 $\text{epsx} = 1.d-6$ ,  $\text{maxitn} = 10000$ ,  $\text{intp} = 500$ .

# Octave-код для двох тестів (початок)

```
# Test 1: ralmprr running time for n = 200 and m = 1000
# Test 2: ralmprr running time for n = 1000 and m = 200

n = 200, m = 1000, # Test 1
#n = 1000, m = 200, # Test 2
rand("seed", 2023); A = 10*rand(m,n);
xstar=round(10.0*rand(n,1)+0.5); y = A*xstar;

alpha = 2.0, h0 = 1.0, q1 = 1.0, epsg = 1.d-16, epsx = 1.d-6,
maxitn = 10000, intp = 500, x0 = round(5.0*rand(n,1));

printf("\n Test 1: ralmprr running time for n = 200 and m = 1000 \n");
#printf("\n Test 2: ralmprr running time for n = 1000 and m = 200 \n");

pip2Lambda = [ 1.0 1.0 1.0; 1.0 1.0 0.0; 1.0 2.0 1.0; 1.0 2.0 0.0;
2.0 1.0 1.0; 2.0 1.0 0.0; 2.0 2.0 1.0; 2.0 2.0 0.0];
```

## Octave-код для двох тестів (продовження)

```

table = [];
for i = 1:rows(p1p2Lambda)
    p1 = p1p2Lambda(i,1), p2 = p1p2Lambda(i,2), lambda = p1p2Lambda(i,3),
    time0 = time();
    [xp,fp,itn,nfg,ist] = ralmpr(A,y,p1,lambda,p2,x0,alpha,h0,q1,eps_g,eps_x,maxitn,intp);
    fp, itn, nfg, ist,
    time1 = time() - time0,
    dx = norm(xp-xstar);
    na1 = 0; eps_na1 = 0.1;
    na2 = 0; eps_na2 = 0.01;
    for i = 1:n
        if (abs(xp(i))>eps_na1) na1 = na1 +1; endif
        if (abs(xp(i))>eps_na2) na2 = na2 +1; endif
    endfor
    table = [table; p1 p2 lambda time1 itn nfg na1 na2 fp dx];
    n, na1, na2, m,
    temp = A*xp - y; sumtemp = sum(abs(temp)),
    sumxp = sum(abs(xp)),
endfor

printf(" epsx = %6.1e \n", epsx);
printf(" p1 p2 lambda time itn nfg na1 na2");
printf(" fp dx \n");
for (i = 1:rows(table))
    printf(" %4.1f %4.1f %4.1f %4.2f %6d %5d %4d %4d %10.5e %10.1e\n", table(i,1:10))
endfor

```

## Результати експериментів для двох тестів

Перший тест ( $n = 200$ ,  $m = 1000$ )

p1	p2	lambda	time	itn	nfg	na1	na2	fp	dx
1.0	1.0	1.0	6.99	4447	4497	200	200	1.12400e+03	4.3e-07
1.0	1.0	0.0	7.05	4445	4495	200	200	9.25158e-04	4.2e-07
1.0	2.0	1.0	7.27	4444	4494	200	200	8.02400e+03	4.4e-07
1.0	2.0	0.0	7.13	4445	4495	200	200	9.25158e-04	4.2e-07
2.0	1.0	1.0	8.47	4352	5291	200	200	1.12400e+03	2.2e-05
2.0	1.0	0.0	8.37	4342	5319	200	200	5.04195e-10	2.8e-07
2.0	2.0	1.0	8.51	4366	5396	200	200	8.02375e+03	6.8e-03
2.0	2.0	0.0	8.54	4342	5319	200	200	5.04195e-10	2.8e-07

Другий тест ( $n = 1000$ ,  $m = 200$ )

p1	p2	lambda	time	itn	nfg	na1	na2	fp	dx
1.0	1.0	1.0	80.08	10000	10057	199	232	5.17858e+03	4.6e+02
1.0	1.0	0.0	36.31	4524	4580	1000	1000	6.82137e-04	9.4e+01
1.0	2.0	1.0	82.72	10000	11410	1000	1000	3.15065e+04	8.4e+01
1.0	2.0	0.0	36.06	4524	4580	1000	1000	6.82137e-04	9.4e+01
2.0	1.0	1.0	80.55	10000	10247	199	209	5.17852e+03	4.6e+02
2.0	1.0	0.0	36.95	4372	5333	1000	1000	3.26497e-10	9.4e+01
2.0	2.0	1.0	83.72	10000	12427	1000	1000	3.15062e+04	8.4e+01
2.0	2.0	0.0	36.85	4372	5333	1000	1000	3.26497e-10	9.4e+01

Тестові задачі програма **ralmpr** вирішує за 7–8 секунд, якщо  $n=200$  та  $m=1000$ , за 80–83 секунди, якщо  $n=1000$  та  $m=200$ .  
Процесор Intel Core i5-9400f, 2.9 GHz, 16GB RAM, Octave 5.1.0.

# Висновки:

Для знаходження параметрів лінійної регресії за критерієм найменших модулів в степені  $p_1$  з регуляризацією параметрів за критерієм найменших модулів в степені  $p_2$  розроблено Octave-програму **ralmpr** (**r**-algorithm for least **m**oduli to the power of **p** with **r**egularization), де  $1 \leq p_1, p_2 \leq 2$ .

Програма **ralmpr** дозволяє налаштовувати степені  $p_1$  та  $p_2$  на пошук параметрів лінійної регресії за правилами ridge-регресії ( $p_1=p_2=2$ ) та lasso-регресії ( $p_1=2, p_2=1$ ). Окрім цього, програма дозволяє знаходити найкращі параметри лінійної регресії, які забезпечуються використанням інших значень степенів  $p_1, p_2 \in [1, 2]$ .

# Подяки

Роботу підтримано грантами  
Національного фонду досліджень України,  
№ 2021.01/0136,  
Volkswagen Foundation, № 97775

# Запитання?

ДЯКУЮ ЗА УВАГУ!