r-Алгоритм для навчання моделей лінійної регресії

Стецюк П.І., Хом'як О.М.

Міжнародна науково-практична конференція
"ЦИФРОВІ ТЕХНОЛОГІЇ В ЕНЕРГЕТИЦІ І АВТОМАТИЦІ"

7 червня 2024, Київ

План доповіді

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

План доповіді

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 $g_f(x_k)$.

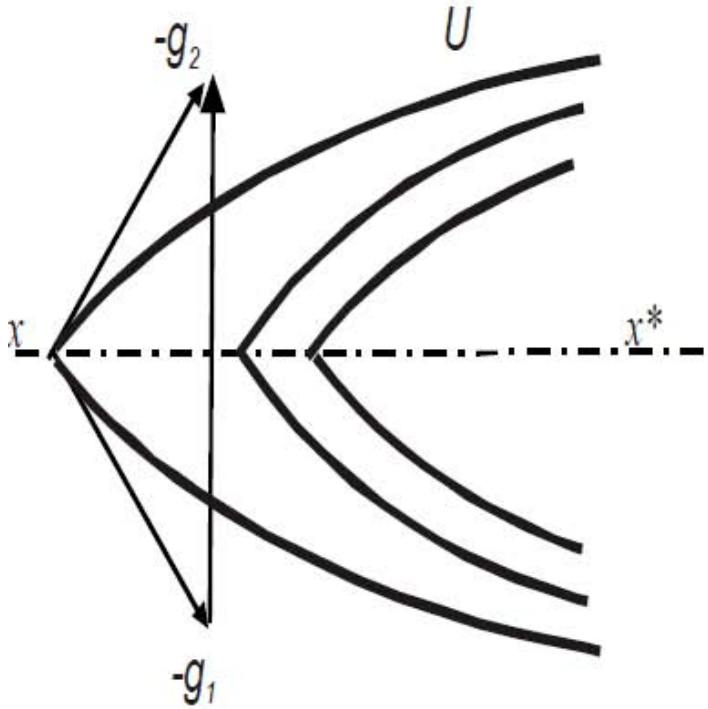
Вони отримали таку назву від слова "різниця".

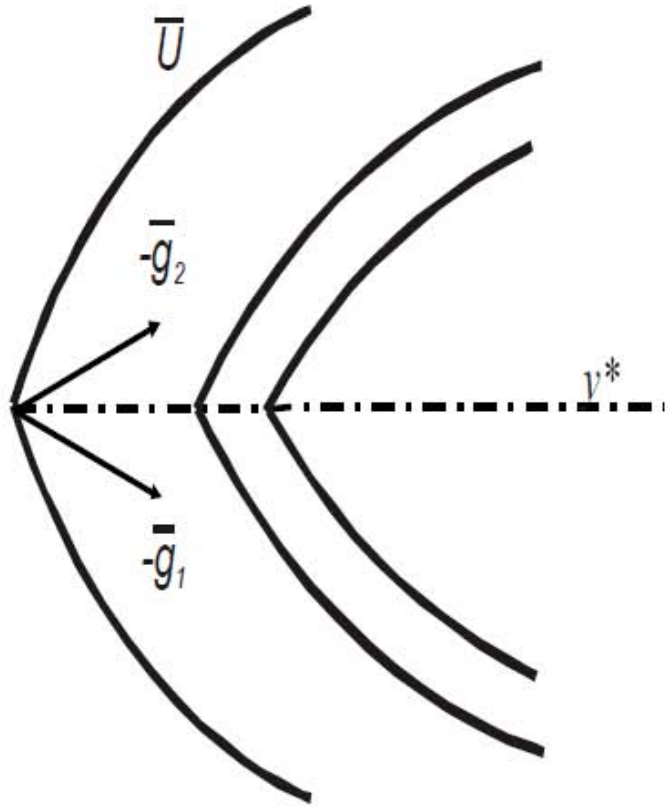
Їх називають r-алгоритмами Шора (1970), або r-алгоритмами Шора–Журбенко (1971).


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:

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

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

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

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 η 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: ralgb5 (2011, Стецюк П.І.), ralgb4 (2016), ralgb5a (2017, грудень 2020) – спрощена версія ralgb5.

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 ralgb5a (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 ralgb5a

Для мінімізації негладких функцій рекомендується:

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

Якщо відома оцінка відстані від стартової точки x_0 до точки мінімуму x^*, то початковий крок h_0 доцільно вибирати

$$h_0 \approx \|x_0 - x^*\|.$$

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

Модель лінійної регресії ($m > n$)

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

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

де a_{ij} – відомі коефіцієнти, u_i – невідомі випадкові величини.

Рівняння (4) можна записати в матричній формі:

$$y = Ax + u, \tag{4a}$$

де $y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$, $u = (u_1, \ldots, u_m)^T \in \mathbb{R}^m$, A – $m \times n$-матриця, $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ – 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:

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

де функція $f_p(x)$ є негладкою, якщо $p = 1$, та $f_p(x)$ 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

$$f^*_{LMM} = \min_{x\in\mathbb{R}^n, z\geq 0} \sum_{i=1}^{m} z_i \tag{6}$$

при обмеженнях

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

Для МНК маємо безумовну задачу мінімізації: знайти

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

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

Узагальнена лінійна регресія $(1 \leq p, q \leq 2)$

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

$$f_{pq}(x^*_{pq}) = \min_{x \in \mathbb{R}^n} \left\{ f_{pq}(x) = \sum_{i=1}^{m} \left| y_i - \sum_{j=1}^{n} a_{ij} x_j \right|^p + \lambda \sum_{i=1}^{n} \left| x_i \right|^q \right\}, \quad (9)$$

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

Функція $f_{pq}(x)$ є негладкою, якщо $p = 1$ або $q = 1$, та $f_{pq}(x)$ є гладкою, якщо $p > 1$ та $q > 1$.

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 $f_{pq}(x)$ в задачі (9)

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

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

Octave-функція ralmpr

Алгоритм знаходження наближення до точки x^*_{pq} в задачі (9) реалізовано octave програмою **ralmpr** (**r**-**a**lgorithm for **l**east **m**oduli to the power of **p** with **r**egularization), код якої є нижче.

```
# Вхідні параметри:
#   A, y, p, lambda, q - дані про задачу (9), 1 <= p, q <= 2
#   x0(n,1), alpha, h0, q1 - стартова точка та параметри r-алгоритму
#   epsx, epsg, maxitn, intp - параметри зупинки та інтервал друку
# Вихідні параметри:
#   xpq(n,1), fpq - 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 [xpq,fpq,itn,nfg,ist] = ralmpr(A,y,p,lambda,q,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 = fpq = sum(abs(temp).^p);         #row03
g0 = p*A'*(sign(temp).*(abs(temp)).^(p-1)); nfg = 1;                #row04
fr = fpq = fpq + lambda*sum(abs(x).^q);                             #row05
g0 = g0 + lambda*q*(sign(x).*(abs(x)).^(q-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)                                                    #row11
   dx = B * (g1 = B' * g0)/norm(g1);                                    #row12
   d = 1; ls = 0; ddx = 0;                                              #row12
   while (d > 0)                                                        #row13
      x -= hs * dx; ddx += hs * norm(dx);                               #row14
      temp = A*x - y; f = sum(abs(temp).^p);                            #row15
      g1 = p*A'*(sign(temp).*(abs(temp)).^(p-1));                       #row16
      f = f + lambda*sum(abs(x).^q);                                    #row17
      g1 = g1 + lambda*q*(sign(x).*(abs(x)).^(q-1));                    #row18
      nfg ++;                                                           #row19
      if (f < fr)  fr = fpq = f; xr = xpq = x;  endif                   #row20
      if(norm(g1) < epsg) ist = 2;  return; endif                       #row21
      ls ++; (mod(ls,3) == 0) && (hs *= 1.1);                           #row22
      if(ls > 500) ist = 5;  return; endif                              #row23
      d = dx' * g1;                                                     #row24
   endwhile                                                             #row25
   (ls == 1) && (hs *= q1); lsa=lsa+ls; lsm=max(lsm,ls);                #row26
   if(mod(itn,intp)==0)                                                 #row27
     if (intp>0)                                                        #row28
       printf("itn %4d  f %14.6e  fr %14.6e", itn, f, fr);              #row29
       printf("  nfg %4d  lsa %3d  lsm %3d\n", nfg, lsa, lsm);          #row30
     endif                                                              #row31
     lsa=0; lsm=0;                                                      #row32
   endif                                                                #row33
   if(ddx < epsx) ist = 3;  return; endif                               #row34
   xi = (dg = B' * (g1 - g0) )/norm(dg);                                #row35
   B += (1 / alpha - 1) * B * xi * xi';                                 #row36
   g0 = g1;                                                             #row37
endfor                                                                  #row38
ist = 4;                                                                #row39
endfunction                                                             #row40
```

План доповіді

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*rand(m,n)**, **y = A*xstar(n,1)**, **xstar(n,1)=round(10*rand(n,1)+0.5)**;
стартова точка **x0(n,1)=round(5*rand(n,1))**;
параметри r-алгоритму:
alpha = 2.0, h0 = 1.0, q1 = 1.0, epsg = 1.d-16, epsx = 1.d-6, maxitn = 10000, intp = 500.

Octave-код для двох тестів (початок)

```
# Test 1: ralmpr runnning time for n = 200 and m = 1000
# Test 2: ralmpr runnning time for n = 1000 and m = 200

n = 200, m = 1000,    # Test 1
#n = 1000, m = 200,    # Test 2
rand("seed", 2024); 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: ralmpr runnning time for n = 200 and m = 1000 \n");
#printf("\n Test 2: ralmpr runnning time for n = 1000 and m = 200 \n");

pqLambda = [ 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(pqLambda)
   p = pqLambda(i,1), q = pqLambda(i,2), lambda = pqLambda(i,3),
   time0 = time();
   [xpq,fpq,itn,nfg,ist] = ralmpr(A,y,p,lambda,q,x0,alpha,h0,q1,epsg,epsx,maxitn,intp);
   fpq, itn, nfg, ist,
   time1 = time() - time0,
   dx = norm(xpq-xstar);
   na1 = 0; eps_na1 = 0.1;
   na2 = 0; eps_na2 = 0.01;
   for i = 1:n
      if (abs(xpq(i))>eps_na1) na1 = na1 +1; endif
      if (abs(xpq(i))>eps_na2) na2 = na2 +1; endif
   endfor
   table = [table; p q lambda time1 itn nfg na1 na2 fpq dx];
   n, na1, na2, m,
   temp = A*xpq - y; sumtemp = sum(abs(temp)),
   sumxpq = sum(abs(xpq)),
endfor

printf(" epsx = %6.1e \n", epsx);
printf("   p     q   lambda  time    itn   nfg   na1   na2");
printf("      fpq           dx  \n");
for (i = 1:rows(table))
   printf(" %4.1f  %4.1f  %4.1f   %5.2f %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)

p	q	lambda	time	itn	nfg	na1	na2	fpq	dx
1.0	1.0	1.0	7.87	4441	4489	200	200	1.13800e+03	5.2e-07
1.0	1.0	0.0	7.81	4442	4490	200	200	9.92370e-04	5.1e-07
1.0	2.0	1.0	7.55	4446	4494	200	200	7.93400e+03	4.5e-07
1.0	2.0	0.0	7.05	4442	4490	200	200	9.92370e-04	5.1e-07
2.0	1.0	1.0	8.74	4342	5346	200	200	1.13800e+03	2.1e-05
2.0	1.0	0.0	8.76	4358	5333	200	200	4.70285e-10	2.8e-07
2.0	2.0	1.0	8.71	4352	5352	200	200	7.93378e+03	6.3e-03
2.0	2.0	0.0	8.74	4358	5333	200	200	4.70285e-10	2.8e-07

Другий тест (n = 1000, m = 200)

p	q	lambda	time	itn	nfg	na1	na2	fpq	dx
1.0	1.0	1.0	82.68	10000	10059	200	236	5.20590e+03	4.7e+02
1.0	1.0	0.0	37.00	4511	4569	1000	1000	6.78320e-04	9.2e+01
1.0	2.0	1.0	84.17	10000	11453	1000	1000	3.14436e+04	8.2e+01
1.0	2.0	0.0	36.99	4511	4569	1000	1000	6.78320e-04	9.2e+01
2.0	1.0	1.0	82.30	10000	10234	200	219	5.20584e+03	4.8e+02
2.0	1.0	0.0	38.25	4467	5468	1000	1000	3.66193e-10	9.2e+01
2.0	2.0	1.0	85.49	10000	12254	1000	1000	3.14434e+04	8.2e+01
2.0	2.0	0.0	38.16	4467	5468	1000	1000	3.66193e-10	9.2e+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 з регуляризацією параметрів за критерієм найменших модулів в степені q розроблено Octave-програму **ralmpr** (**r**-**a**lgorithm for **l**east **m**oduli to the power of **p** with **r**egularization), де $1 \leq p, q \leq 2$.

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

Подяки

Роботу підтримано грантами

Національного фонду досліджень України, № 2021.01/0136,

Volkswagen Foundation, № 97775

Запитання?

ДЯКУЮ ЗА УВАГУ!