User's Guide for NNLSSPAR

Fatih S. AKTAŞ, Ömer Ekmekcioğlu and Mustafa Ç. Pınar

Nov 2020

This guide explains how to use NNLSSPAR, the python package based on the paper;

Fatih S. Aktaş, Ömer Ekmekcioğlu and Mustafa Ç. Pınar, "Provably Optimal Sparse Solutions to Over-determined Linear Systems with Non-Negativity Constraints in a Least-Squares Sense by Implicit Enumeration"

1 Problem

NNLSSPAR solves the following problem;

$$\begin{array}{ll} \min\limits_{x \in R^n} & \|Ax - b\|_2^2 \\ \text{s.t.} & x \geq 0 \\ & \|x\|_0 \leq s \end{array}$$

2 Details of Parameters and Variables

In this section, we explain the parameters of NNLSSPAR library.

Parameters:

1. A (input) = $m \times n$ matrix storing the values of the independent variables where m must be greater than n
2. b (input) = $m \times 1$ vector storing the values of the dependent variables
3. s (input) = An integer value indicating sparsity level (maximum number of nonzero elements in the solution)
4. out (input, by default it is 1) = An integer value, that is a parameter controlling the detail level of output:
   (a) out = 0, Nothing will be printed

(b) out = 1, The results and the hardware usage of the algorithm will be printed

(c) out = 2, At every iteration, lower bound of the current node and the number of nodes searched so far will be printed. After the algorithm terminates NNLSSPAR will report the results and the hardware usage. Although it is good to observe the progress of the algorithm, it should be noted here that it slows down the algorithm significantly.

5. C (input, by default it is $\emptyset$) = Array of integers, storing the indices of chosen variables

6. many (input, by default it is 4) = An integer specifying number of best subsets to be found

7. residual_squared (output) = a list of length *many* ($s \times many$ if all subsets function is used) containing residuals corresponding to subset found for some sparsity level. This only shows the residuals of the system after QR decomposition. True residual squared of the systems are residual_squared + permanent_residual

8. indexes (output) = a list of length *many* ($s \times many$ if all subsets function is used) containing indices of the independent variables that make the subsets for some sparsity level

9. coefficients (output) = a list of length *many* ($s \times many$ if all subsets function is used) containing coefficients of the least squares solution of each subset for some sparsity level

10. permanent_residual (output) = a float that records the residual squared of the system after orthogonal transform using QR decomposition of the matrix A

11. cpu (output) = a float showing the CPU time of the algorithm in seconds. CPU time of the algorithm is print if out is not 0

12. memory (output) = a float showing the total memory usage of the algorithm in bytes. Detailed table of memory usage is print if out is not 0

13. real (output) = a float showing the wall time of the algorithm in seconds.

14. nodes (hidden) = a list of integers showing number of visited nodes in the graph. It is equal to the number of branching done due to best first search if only one solution is found. For each sparsity level it will be reported separately.

15. rem_qsize (hidden) = a list of integers showing the number of unvisited nodes in the graph. This will be equal to nodes if only one solution is found. For each sparsity level it will be reported separately.

3 How to use NNLSSPAR

Suppose A and b are generated as follows;

A = numpy.random.normal(0,1,[20,10])
x = numpy.reshape([3,0,0,2,1,0,0,4,0,0],[10,1])
b = A.dot(x) + numpy.random.normal(0,1,[20,1])
s = 4

First initialize the NNLSSPAR object;

u = NNLSSPAR(A,b,s)

3.1 Original Problem

We can solve the problem described in section 1 by calling the function;

sol = u.solve()

Algorithm should output something like this;

CPU time of the algorithm 0.0 seconds
Wall time of the algorithm 0.0009982585906982422 seconds
Partition of a set of 20 objects. Total size = 1984 bytes.

Index	Count	%	Size	%	Cumulative	%	Kind (class / dict of class)
0	2	10	480	24	480	24	dict (no owner)
1	1	5	416	21	896	45	types.FrameType
2	2	10	296	15	1192	60	list
3	7	35	168	8	1360	69	float
4	1	5	128	6	1488	75	numpy.ndarray
5	2	10	128	6	1616	81	types.MethodType
6	1	5	120	6	1736	88	asyncio.events.TimerHandle
7	1	5	88	4	1824	92	functools.partial
8	1	5	72	4	1896	96	builtins.Context
9	1	5	56	3	1952	98	tuple
10	1	5	32	2	1984	100	numpy.float64

The answer is registered to the object attributes. Calling the appropriate attributes of using the attributes;

The Subset Indices
u.indices
[[7, 0, 3, 4]]
Coefficients of the variables in the order given by indices

u.coefficients
[array([3.7738094 , 2.98367905, 1.81056504, 1.40309046])]
Residual squared without the QR shrinking step
u.residual_squared
[4.3938147543739525]
Actual Residual Squared of the system
u.residual_squared + u.permanent_residual
[9.45157599]
Number of nodes searched
u.nodes
[5]

3.2 Multiple Subsets

Solve function will by default find only one the optimal solution. However, if instead of only the best subset, best k subsets are desired, where $k \leq \binom{n}{s}$, the following modification should be made prior to calling the function.

u.many = 4 , the default value is 1
sol = u.solve()

As before, algorithm will again first output the cpu and wall time, and memory usage.

CPU time of the algorithm 0.0 seconds
Wall time of the algorithm 0.001995563507080078 seconds
Partition of a set of 28 objects. Total size = 2512 bytes.

Index	Count	%	Size	%	Cumulative	%	Kind (class / dict of class)
0	5	18	584	23	584	23	list
1	4	14	512	20	1096	44	numpy.ndarray
2	1	4	416	17	1512	60	types.FrameType
3	1	4	240	10	1752	70	dict (no owner)
4	7	25	168	7	1920	76	float
5	4	14	128	5	2048	82	numpy.float64
6	2	7	128	5	2176	87	types.MethodType
7	1	4	120	5	2296	91	asyncio.events.TimerHandle
8	1	4	88	4	2384	95	functools.partial
9	1	4	72	3	2456	98	builtins.Context
10	1 6 4	56	2	2512	100	tuple	

Solutions are given in the order given by their residual.

The Subset Indices
u.indices
[[7, 0, 3, 4], [7, 0, 3, 5], [7, 0, 3, 2], [7, 0, 3, 6]]
Coefficients of the variables in the order given by indices
u.coefficients
[array([3.82789364, 2.64871044, 1.94850681, 0.62713298]), array([4.05996493, 2.45152795, 1.66674643, 0.3997306]), array([3.91392406, 2.4847922 , 1.73656329, 0.30950597]), array([4.0250534 , 2.51628932, 1.71440157, 0.20686085])]
Residual squared without the QR shrinking step
u.residual_squared
[4.977293002824664, 11.134415166817956, 12.541409437782544, 13.440774669375482]
Actual Residual Squared of the system
u.residual_squared + u.permanent_residual
[11.46946975 17.62659191 19.03358618 19.93295141]
Number of nodes searched
u.nodes
[11]

3.3 All Subsets

If best subsets for i = 1, 2 , .. , s-1, s and at each level best k subsets are desired, where $k \leq n$ instead of solving them separately, the following function can be called;

u.solve_allsubsets()

Which will again, first print out cpu and wall time, and memory usage of the algorithm. Output will be very crowded but using the attributes in a similar fashion given in Section 3.2, answers can be obtained.

4 Changing Defaults

4.1 Enforcing Variables

If prior to solving a problem, particular variables are known to be very likely to be in the solution set, or a solution where those particular variables are in the solution set is desired, they can be given as an input while calling the function. However, this is not allowed for all subsets problem.

enforce_indexes = [0,1]
u.solve(enforce_indexes)

The Subset Indices
u.indices

[[0, 1, 7, 3]]
Coefficients of the variables in the order given by indices
u.coefficients
[array([3.36168499, 0.38791543, 4.33968551, 1.82235801])]
Residual squared without the QR shrinking step
u.residual_squared
[16.877649484000948]
Actual Residual Squared of the system
u.residual_squared + u.permanent_residual
[23.48006878]
Number of nodes searched
u.nodes
[3]

If an invalid or illogical set of indexes are given, they will be ignored and error message will be print.

5 Extended Least Squares

NNLSSPAR also has a function to solve the following problem;

$$\begin{aligned} \min_{x,y \in R^n} \quad & \|Ax + By - b\|_2^2 \\ \text{s.t.} \quad & x \geq 0 \\ & \|x\|_0 \leq s \end{aligned}$$

We call this problem extended least squares because there is a variable independent of cardinality and non-negativity constraint. If we define matrix C and vector z as follows;

$$C = \begin{pmatrix} A & B \end{pmatrix} z = \begin{bmatrix} x \\ y \end{bmatrix}$$

Then the new objective function looks like the following;

$$\min_{z \in R^n} \|Cz - b\|_2^2$$

Then columns, variables, over which the cardinality and non-negativity constraint will not be enforced can be given to the NNLSSPAR.

extend = [7,8,9]

u.solve_elsq(extend)
The Subset Indices
u.indices
[[0, 3, 4, 2]]

Coefficients of the variables in the order given by indices
u.coefficients
[array([3.0360357 , 2.01682858, 0.96026743, 0.89119861, 3.85052488, 0.01508911,
-0.00817506])]
Residual squared without the QR shrinking step
u.residual_squared
[0.916867064487307]
Actual Residual Squared of the system
u.residual_squared + u.permanent_residual
[7.88937791]
Number of nodes searched
u.nodes
[5]

The coefficients are given in the following structure;

[coefficients of constrained variables in the order given by indices, coefficients of extended variables by the given order]