NAG Toolbox

nag_best_subset_given_size_revcomm (h05aa)

1 Purpose

Given a set of m features and a scoring mechanism for any subset of those features, nag_best_subset_given_size_revcomm (h05aa) selects the best n subsets of size p using a reverse communication branch and bound algorithm.

2 Syntax

```
[irevcm, drop, lz, z, la, a, bscore, bz, icomm, rcomm, ifail] =
nag_best_subset_given_size_revcomm(irevcm, mincr, m, ip, drop, lz, z, la, a,
bscore, bz, mincnt, gamma, acc, icomm, rcomm, 'nbest', nbest, 'licomm', licomm,
'lrcomm', lrcomm)
```

```
[irevcm, drop, lz, z, la, a, bscore, bz, icomm, rcomm, ifail] = h05aa(irevcm,
mincr, m, ip, drop, lz, z, la, a, bscore, bz, mincnt, gamma, acc, icomm, rcomm,
'nbest', nbest, 'licomm', licomm, 'lrcomm', lrcomm)
```

3 Description

Given $\Omega = \{x_i : i \in \mathbb{Z}, 1 \leq i \leq m\}$, a set of m unique features and a scoring mechanism $f(S)$ defined for all $S \subseteq \Omega$ then nag_best_subset_given_size_revcomm (h05aa) is designed to find $S_{o1} \subseteq \Omega, |S_{o1}| = p$, an optimal subset of size p. Here $|S_{o1}|$ denotes the cardinality of S_{o1}, the number of elements in the set.

The definition of the optimal subset depends on the properties of the scoring mechanism, if

$$f(S_i) \leq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \tag{1}$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\operatorname{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

else if

$$f(S_i) \geq f(S_j), \quad \text{for all } S_j \subseteq \Omega \text{ and } S_i \subseteq S_j \tag{2}$$

then the optimal subset is defined as one of the solutions to

$$\underset{S \subseteq \Omega}{\operatorname{minimize}} f(S) \quad \text{subject to} \quad |S| = p.$$

If neither of these properties hold then nag_best_subset_given_size_revcomm (h05aa) cannot be used.

As well as returning the optimal subset, S_{o1}, nag_best_subset_given_size_revcomm (h05aa) can return the best n solutions of size p. If S_{oi} denotes the ith best subset, for $i = 1, 2, \ldots, n-1$, then the $(i+1)$th best subset is defined as the solution to either

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\operatorname{maximize}} f(S) \quad \text{subject to} \quad |S| = p$$

or

$$\underset{S \subseteq \Omega - \{S_{oj} : j \in \mathbb{Z}, 1 \leq j \leq i\}}{\operatorname{minimize}} f(S) \quad \text{subject to} \quad |S| = p$$

depending on the properties of f.

The solutions are found using a branch and bound method, where each node of the tree is a subset of Ω. Assuming that (1) holds then a particular node, defined by subset S_i, can be trimmed from the tree if $f(S_i) < \hat{f}(S_{on})$ where $\hat{f}(S_{on})$ is the nth highest score we have observed so far for a subset of size p, i.

e., our current best guess of the score for the nth best subset. In addition, because of (1) we can also drop all nodes defined by any subset S_j where $S_j \subseteq S_i$, thus avoiding the need to enumerate the whole tree. Similar short cuts can be taken if (2) holds. A full description of this branch and bound algorithm can be found in Ridout (1988).

Rather than calculate the score at a given node of the tree nag_best_subset_given_size_revcomm (h05aa) utilizes the fast branch and bound algorithm of Somol *et al.* (2004), and attempts to estimate the score where possible. For each feature, x_i, two values are stored, a count c_i and $\hat{\mu}_i$, an estimate of the contribution of that feature. An initial value of zero is used for both c_i and $\hat{\mu}_i$. At any stage of the algorithm where both $f(S)$ and $f(S - \{x_i\})$ have been calculated (as opposed to estimated), the estimated contribution of the feature x_i is updated to

$$\frac{c_i\hat{\mu}_i + [f(S) - f(S - \{x_j\})]}{c_i + 1}$$

and c_i is incremented by 1, therefore at each stage $\hat{\mu}_i$ is the mean contribution of x_i observed so far and c_i is the number of observations used to calculate that mean.

As long as $c_i \geq k$, for the user-supplied constant k, then rather than calculating $f(S - \{x_\mathrm{i}\})$ this function estimates it using $\hat{f}(S - \{x_\mathrm{i}\}) = f(S) - \gamma\hat{\mu}_i$ or $\hat{f}(S) - \gamma\hat{\mu}_i$ if $f(S)$ has been estimated, where γ is a user-supplied scaling factor. An estimated score is never used to trim a node or returned as the optimal score.

Setting $k = 0$ in this function will cause the algorithm to always calculate the scores, returning to the branch and bound algorithm of Ridout (1988). In most cases it is preferable to use the fast branch and bound algorithm, by setting $k > 0$, unless the score function is iterative in nature, i.e., $f(S)$ must have been calculated before $f(S - \{x_i\})$ can be calculated.

4 References

Narendra P M and Fukunaga K (1977) A branch and bound algorithm for feature subset selection *IEEE Transactions on Computers* **9** 917–922

Ridout M S (1988) Algorithm AS 233: An improved branch and bound algorithm for feature subset selection *Journal of the Royal Statistics Society, Series C (Applied Statistics) (Volume 37)* **1** 139–147

Somol P, Pudil P and Kittler J (2004) Fast branch and bound algorithms for optimal feature selection *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume 26)* **7** 900–912

5 Parameters

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than bscore must remain unchanged**.

5.1 Compulsory Input Parameters

1: **irevcm** – INTEGER

On initial entry: must be set to 0.

On intermediate re-entry: **irevcm** must remain unchanged.

Constraint: **irevcm** $= 0$ or 1.

2: **mincr** – INTEGER

Flag indicating whether the scoring function f is increasing or decreasing.

mincr $= 1$
$f(S_i) \leq f(S_j)$, i.e., the subsets with the largest score will be selected.

mincr $= 0$
$f(S_i) \geq f(S_j)$, i.e., the subsets with the smallest score will be selected.

For all $S_j \subseteq \Omega$ and $S_i \subseteq S_j$.

Constraint: **mincr** $= 0$ or 1.

3: **m** – INTEGER

m, the number of features in the full feature set.

Constraint: **m** ≥ 2.

4: **ip** – INTEGER

p, the number of features in the subset of interest.

Constraint: $1 \leq$ **ip** $\leq$ **m**.

5: **drop** – INTEGER

On initial entry: **drop** need not be set.

On intermediate re-entry: **drop** must remain unchanged.

6: **lz** – INTEGER

On initial entry: **lz** need not be set.

On intermediate re-entry: **lz** must remain unchanged.

7: **z**(**m** − **ip**) – INTEGER array

On initial entry: **z** need not be set.

On intermediate re-entry: **z** must remain unchanged.

8: **la** – INTEGER

On initial entry: **la** need not be set.

On intermediate re-entry: **la** must remain unchanged.

9: **a**(**max**(**nbest**, **m**)) – INTEGER array

On initial entry: **a** need not be set.

On intermediate re-entry: **a** must remain unchanged.

10: **bscore**(**max**(**nbest**, **m**)) – REAL (KIND=nag_wp) array

On initial entry: **bscore** need not be set.

On intermediate re-entry: **bscore**(j) must hold the score for the jth subset as described in **irevcm**.

11: **bz**(**m** − **ip**, **nbest**) – INTEGER array

On initial entry: **bz** need not be set.

On intermediate re-entry: **bz** must remain unchanged.

12: **mincnt** – INTEGER

k, the minimum number of times the effect of each feature, x_i, must have been observed before $f(S - \{x_i\})$ is estimated from $f(S)$ as opposed to being calculated directly.

If $k = 0$ then $f(S - \{x_i\})$ is never estimated. If **mincnt** < 0 then k is set to 1.

13: **gamma** – REAL (KIND=nag_wp)

γ, the scaling factor used when estimating scores. If **gamma** < 0 then $\gamma = 1$ is used.

14: **acc**(**2**) – REAL (KIND=nag_wp) array

A measure of the accuracy of the scoring function, f.

Letting $a_i = \epsilon_1 |f(S_i)| + \epsilon_2$, then when confirming whether the scoring function is strictly increasing or decreasing (as described in **mincr**), or when assessing whether a node defined by subset S_i can be trimmed, then any values in the range $f(S_i) \pm a_i$ are treated as being numerically equivalent.

If $0 \leq$ **acc**$(1) \leq 1$ then $\epsilon_1 =$ **acc**(1), otherwise $\epsilon_1 = 0$.

If **acc**$(2) \geq 0$ then $\epsilon_2 =$ **acc**(2), otherwise $\epsilon_2 = 0$.

In most situations setting both ϵ_1 and ϵ_2 to zero should be sufficient. Using a nonzero value, when one is not required, can significantly increase the number of subsets that need to be evaluated.

15: **icomm**(**licomm**) – INTEGER array

On initial entry: **icomm** need not be set.

On intermediate re-entry: **icomm** must remain unchanged.

16: **rcomm**(**lrcomm**) – REAL (KIND=nag_wp) array

On initial entry: **rcomm** need not be set.

On intermediate re-entry: **rcomm** must remain unchanged.

5.2 Optional Input Parameters

1: **nbest** – INTEGER

Default: 1

n, the maximum number of best subsets required. The actual number of subsets returned is given by **la** on final exit. If on final exit **la** $\neq$ **nbest** then **ifail** $= 53$ is returned.

Constraint: **nbest** ≥ 1.

2: **licomm** – INTEGER

Default: the dimension of the array **icomm**.

The length of the array **icomm**. If **licomm** is too small and **licomm** ≥ 2 then **ifail** $= 172$ is returned and the minimum value for **licomm** and **lrcomm** are given by **icomm**(1) and **icomm**(2) respectively.

Constraints:

if **mincnt** $= 0$
, **licomm** $\geq 2 \times \max(\mathbf{nbest}, \mathbf{m}) + \mathbf{m}(\mathbf{m}+2) + (\mathbf{m}+1) \times \max(\mathbf{m} - \mathbf{ip}, 1) + 27$;
otherwise **licomm** $\geq 2 \times \max(\mathbf{nbest}, \mathbf{m}) + \mathbf{m}(\mathbf{m}+3) + (2\mathbf{m}+1) \times \max(\mathbf{m} - \mathbf{ip}, 1) + 25$.

3: **lrcomm** – INTEGER

Default: the dimension of the array **rcomm**.

The length of the array **rcomm**. If **lrcomm** is too small and **licomm** ≥ 2 then **ifail** $= 172$ is returned and the minimum value for **licomm** and **lrcomm** are given by **icomm**(1) and **icomm**(2) respectively.

Constraints:

if **mincnt** $= 0$, **lrcomm** $\geq 9 +$ **nbest** $+$ **m** $\times \max($**m** $-$ **ip**$, 1)$;
otherwise **lrcomm** $\geq 8 +$ **m** $+$ **nbest** $+$ **m** $\times \max($**m** $-$ **ip**$, 1)$.

5.3 Output Parameters

1: **irevcm** – INTEGER

On intermediate exit: **irevcm** $= 1$ and before re-entry the scores associated with **la** subsets must be calculated and returned in **bscore**.

The **la** subsets are constructed as follows:

drop $= 1$
The jth subset is constructed by *dropping* the features specified in the first **lz** elements of **z** and the single feature given in **a**(j) from the full set of features, Ω. The subset will therefore contain **m** $-$ **lz** $- 1$ features.

drop $= 0$
The jth subset is constructed by *adding* the features specified in the first **lz** elements of **z** and the single feature specified in **a**(j) to the empty set, $\emptyset$. The subset will therefore contain **lz** $+ 1$ features.

In both cases the individual features are referenced by the integers 1 to **m** with 1 indicating the first feature, 2 the second, etc., for some arbitrary ordering of the features. The same ordering must be used in all calls to nag_best_subset_given_size_revcomm (h05aa).

If **la** $= 0$, the score for a single subset should be returned. This subset is constructed by adding or removing only those features specified in the first **lz** elements of **z**.

If **lz** $= 0$, this subset will either be Ω or $\emptyset$.

The score associated with the jth subset must be returned in **bscore**(j).

On final exit: **irevcm** $= 0$, and the algorithm has terminated.

2: **drop** – INTEGER

On intermediate exit: flag indicating whether the intermediate subsets should be constructed by dropping features from the full set (**drop** $= 1$) or adding features to the empty set (**drop** $= 0$). See **irevcm** for details.

On final exit: **drop** is undefined.

3: **lz** – INTEGER

On intermediate exit: the number of features stored in **z**.

On final exit: **lz** is undefined.

4: **z**(**m** $-$ **ip**) – INTEGER array

On intermediate exit: **z**(i), for $i = 1, 2, \ldots,$ **lz**, contains the list of features which, along with those specified in **a**, define the subsets whose score is required. See **irevcm** for additional details.

On final exit: **z** is undefined.

5: **la** – INTEGER

On intermediate exit: if **la** > 0, the number of subsets for which a score must be returned.

If **la** $= 0$, the score for a single subset should be returned. See **irevcm** for additional details.

On final exit: the number of best subsets returned.

6: **a**(**max**(**nbest**, **m**)) – INTEGER array

On intermediate exit: **a**(j), for $j = 1, 2, \ldots,$ **la**, contains the list of features which, along with those specified in **z**, define the subsets whose score is required. See **irevcm** for additional details.

On final exit: **a** is undefined.

7: **bscore**(**max**(**nbest**, **m**)) – REAL (KIND=nag_wp) array

On intermediate exit: **bscore** is undefined.

On final exit: holds the score for the **la** best subsets returned in **bz**.

8: **bz**(**m** − **ip**, **nbest**) – INTEGER array

On intermediate exit: **bz** is used for storage between calls to nag_best_subset_given_size_revcomm (h05aa).

On final exit: the jth best subset is constructed by dropping the features specified in **bz**(i, j), for $i = 1, 2, \ldots,$ **m** − **ip** and $j = 1, 2, \ldots,$ **la**, from the set of all features, Ω. The score for the jth best subset is given in **bscore**(j).

9: **icomm**(**licomm**) – INTEGER array

On intermediate exit: **icomm** is used for storage between calls to nag_best_subset_given_size_revcomm (h05aa).

On final exit: **icomm** is not defined. The first two elements, **icomm**(1) and **icomm**(2) contain the minimum required value for **licomm** and **lrcomm** respectively.

10: **rcomm**(**lrcomm**) – REAL (KIND=nag_wp) array

On intermediate exit: **rcomm** is used for storage between calls to nag_best_subset_given_size_revcomm (h05aa).

On final exit: **rcomm** is not defined.

11: **ifail** – INTEGER

On final exit: **ifail** $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 11$

Constraint: **irevcm** $= 0$ or 1.

ifail $= 21$

Constraint: **mincr** $= 0$ or 1.

ifail $= 22$

mincr has changed between calls.

ifail $= 31$

Constraint: **m** ≥ 2.

ifail $= 32$

m has changed between calls.

ifail $= 41$

Constraint: $1 \leq$ **ip** $\leq$ **m**.

ifail $= 42$

ip has changed between calls.

ifail $= 51$

Constraint: **nbest** ≥ 1.

ifail $= 52$

nbest has changed between calls.

ifail $= 53$ (*warning*)

On entry, **nbest** $= \langle$*value*$\rangle$.
But only $\langle$*value*$\rangle$ best subsets could be calculated.

ifail $= 61$

drop has changed between calls.

ifail $= 71$

lz has changed between calls.

ifail $= 91$

la has changed between calls.

ifail $= 111$

bscore$(\langle$*value*$\rangle) = \langle$*value*$\rangle$, which is inconsistent with the score for the parent node.

ifail $= 131$

mincnt has changed between calls.

ifail $= 141$

gamma has changed between calls.

ifail $= 151$

acc(1) has changed between calls.

ifail $= 152$

acc(2) has changed between calls.

ifail $= 161$

icomm has been corrupted between calls.

ifail $= 171$

licomm is too small.
icomm is too small to return the required array sizes.

ifail $= 172$ (*warning*)

Constraint: **licomm** $\geq \langle$*value*$\rangle$.
The minimum required values for **licomm** and **lrcomm** are returned in **icomm**(1) and **icomm**(2) respectively.

ifail $= 181$

rcomm has been corrupted between calls.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The subsets returned by nag_best_subset_given_size_revcomm (h05aa) are guaranteed to be optimal up to the accuracy of your calculated scores.

8 Further Comments

The maximum number of unique subsets of size p from a set of m features is $N = \frac{m!}{(m-p)!p!}$. The efficiency of the branch and bound algorithm implemented in nag_best_subset_given_size_revcomm (h05aa) comes from evaluating subsets at internal nodes of the tree, that is subsets with more than p features, and where possible trimming branches of the tree based on the scores at these internal nodes as described in Narendra and Fukunaga (1977). Because of this it is possible, in some circumstances, for more than N subsets to be evaluated. This will tend to happen when most of the features have a similar effect on the subset score.

If multiple optimal subsets exist with the same score, and **nbest** is too small to return them all, then the choice of which of these optimal subsets is returned is arbitrary.

9 Example

This example finds the three linear regression models, with five variables, that have the smallest residual sums of squares when fitted to a supplied dataset. The data used in this example was simulated.

9.1 Program Text

```
     function h05aa_example

fprintf('h05aa example results\n\n');

% Data required by the scoring function
n      = nag_int(40);
m      = nag_int(14);
[x,y]  = gen_data(n,m);

% Initialize parameters and get communication array lengths
irevcm = nag_int(0);
mincr  = nag_int(0);
ip     = nag_int(5);
drop   = nag_int(0);
lz     = nag_int(0);
mip    = m - ip;
z      = zeros(mip, 1, nag_int_name);
la     = nag_int(0);
nbest  = nag_int(3);
a      = zeros(max(nbest, m), 1, nag_int_name);
bscore = zeros(max(nbest, m), 1);
bz     = zeros(mip, nbest, nag_int_name);
mincnt = nag_int(-1);
```

```
gamma  = -1;
acc    = [0, 0];
icomm = zeros(2, 1, nag_int_name);
rcomm = zeros(0, 0);

warning('off', 'NAG:warning');
[irevcm, drop, lz, z, la, a, bscore, bz, icomm, rcomm, ifail] = ...
    h05aa(...
          irevcm, mincr, m, ip, drop, lz, z, la, a, ...
          bscore, bz, mincnt, gamma, acc, icomm, rcomm);
warning('on', 'NAG:warning');

% Ignore the warning message - required size of communication arrays now
% stored in icomm
rcomm = zeros(icomm(2), 1);
icomm = zeros(icomm(1), 1, nag_int_name);

% Initialization  call
cnt = 0;
[irevcm, drop, lz, z, la, a, bscore, bz, icomm, rcomm, ifail] = ...
    h05aa(...
          irevcm, mincr, m, ip, drop, lz, z, la, a, ...
          bscore, bz, mincnt, gamma, acc, icomm, rcomm);

% Reverse communication loop for best subset routine, terminates on irevcm = 0.
while not(irevcm == 0)
  % Calculate and return the score for the required models and keep track
  % of the number of subsets evaluated
  cnt = cnt + max(1,la);
  [bscore] = calc_subset_score(m, drop, lz, z, la, a, x, y, bscore);

  [irevcm, drop, lz, z, la, a, bscore, bz, icomm, rcomm, ifail] = ...
      h05aa(...
            irevcm, mincr, m, ip, drop, lz, z, la, a, ...
            bscore, bz, mincnt, gamma, acc, icomm, rcomm);
end

% Display the best subsets and corresponding scores.
% h05aa returns a list of features excluded from the best subsets;
% this is inverted to give the set of features included in each subset.
fprintf('\n    Score        Feature Subset\n');
fprintf('    -----        --------------\n');
ibz = 1:m;
for i = 1:la
  mask = ones(1, m, nag_int_name);
  mask(bz(1:mip, i)) = 0;
  fprintf('%12.5e %5d %5d %5d %5d %5d\n', bscore(i), ibz(logical(mask)));
end

fprintf('\n%d subsets evaluated in total\n', cnt);

function [bscore] = calc_subset_score(m, drop, lz, z, la, a, x, y, bscore)
  % Set up the initial feature set.
  % If drop = 0, this is the Null set (i.e. no features).
  % If drop = 1 then this is the full set (i.e. all features)
  if drop == 0
    isx = zeros(m, 1, nag_int_name);
  else
    isx = ones(m, 1, nag_int_name);
  end

  % Add (if drop = 0) or remove (if drop = 1) all the features specified in z
  inv_drop = not(drop);
  isx(z(1:lz)) = inv_drop;

  for i=1:max(la, 1)
    if (la > 0)
      if (i > 1)
        % Reset the feature altered at the last iteration
        isx(a(i-1)) = drop;
      end
      %  Add or drop the i'th feature in a
      isx(a(i)) = inv_drop;
    end

    ip = nag_int(sum(isx));

    % Fit the regression model
    rss = g02da('z', x, isx, ip, y);

    % Return the score (the residual sums of squares)
    bscore(i) = rss;
  end

function [x, y] = gen_data(n,m)
  x = zeros(n,m);
  genid = nag_int(3);
  subid = nag_int(1);
  seed(1) = nag_int(23124124);
  [state, ifail] = g05kf(genid, subid, seed);
  for i = 1:m
    [state, x(1:n,i), ifail] = g05sk( ...
                                      n, 0, sqrt(3), state);
  end
  [state, b, ifail] = g05sk(m, 1.5, 3, state);
  [state, y, ifail] = g05sk(n, 0, 1, state);
  y = x*b + y;
```

9.2 Program Results

```
     h05aa example results

    Score        Feature Subset
    -----        --------------
 1.21567e+03     3     4     6     7    14
 1.32495e+03     3     5     6     7    14
 1.37244e+03     3     5     6    12    14

337 subsets evaluated in total
```