



“Nuts and Bolts”

Sections:

1. Read in the data
2. Essential settings
3. Common settings
4. Special purpose settings
5. Output control

1. Read in the data

Go to line 250:

```
open(16, file='data.train', status='old')
do n=1,nsample0
    read(16,*) (x(m,n),m=1,mdim),cl(n)
enddo
close(16)
if(ntest.gt.1) then
    open(17, file='data.test', status='old')
```

Control parameters

```
c      DESCRIBE DATA
  1      mdim=36, nsample0=4435, nclass=6, maxcat=1,
  1      ntest=0, labelts=0, labeltr=1,
c
c      SET RUN PARAMETERS
  2      mtry0=6, ndsize=1, jbt=100, look=100, lookcls=1,
  2      jclasswt=0, mdim2nd=0, mselect=0, iseed=4351,
c
c      SET IMPORTANCE OPTIONS
  3      imp=0, interact=0, impn=0, impfast=0,
c
c      SET PROXIMITY COMPUTATIONS
  4      nprox=0, nrnn=200,
c
c      SET OPTIONS BASED ON PROXIMITIES
  5      noutlier=0, nscale=0, nprot=0,
c
c      REPLACE MISSING VALUES
  6      code=-999, missfill=0, mfixrep=0,
c
c      GRAPHICS
  7      iviz=0,
c
c      SAVING A FOREST
  8      isaverf=0, isavepar=0, isavefill=0, isaveprox=0,
c
c      RUNNING A SAVED FOREST
  9      irunrf=0, ireadpar=0, ireadfill=0, ireadprox=0)
```

2. Essential settings

`mdim` = number of input variables

`nsample0` = training set size

`jbt` = number of trees in the forest

`mtry0` = number of random splits

Choosing m_{try}

- Start with $m_{try} = \sqrt{m_{dim}}$
- Try doubling and halving m_{try}
- If oob error rate is better with large (small) m_{try} , go larger (smaller) until the error rate starts to increase

Does the training set
have class labels?

yes

no

labeltr = 1

labeltr = 0

nclass = number of classes

nclass = 2

*unsupervised
learning*

Is there a test set?

yes

no

$n_{\text{test}} = \text{test set size}$

$n_{\text{test}} = 0$

$\text{labelts} = 1$ if labeled

$\text{labelts} = 0$

$\text{labelts} = 0$ if unlabeled

Are there categorical
input variables?

yes

no

`maxcat` = max number of categories

`maxcat` = 1

Go to line 280:

```
c   SET CATEGORICAL VALUES
    do m=1,mdim
        cat(m)=1
    enddo
```

3. Common settings

- Importance
- Class weights
- Proximities, scaling, prototypes, outliers
- Missing values
- Graphics

Importance

- $\text{imp} = 0 \rightarrow \text{imp} = 1$ for variable importance
- $\text{mdim2nd} = m$ does a second run with m most important variables (requires $\text{imp} = 1$)

Class weights

- `jclasswt = 0` equal class weights
- `jclasswt = 1` unequal class weights

Go to line 290:

```
c      SET CLASS WEIGHTS
      do j=1,nclass
          classwt(j)=1
      enddo
```

note: setting class weights may require inspection of the oob error rate for each class, and suitable adjustment of the class weights

Proximities, scaling, prototypes, outliers

- `nprox = 0` no proximities
- `nprox = 1` training set proximities
- `nprox = 2` training and test set proximities

The following options require `nprox > 0`:

- `noutlier = 1` outlier detection (2 adds test set)
- `nscale` = number of MDS variables
- `nprot` = number of prototypes for each class

Choosing `nrnn`

`nrnn` = the number of nearest neighbors for which to compute proximities

Do you want prototypes?

```
graph TD; Q("Do you want prototypes?") -- yes --> A["set nrnn = 1/3 to 1/2 the size of the largest class"]; Q -- no --> B["set nrnn using computational considerations"];
```

yes

set `nrnn` = $1/3$
to $1/2$ the size
of the largest
class

no

set `nrnn` using
computational
considerations

Missing values

- `missfill = 1` does a fast, rough missing value replacement
- `missfill = 2` does a careful missing value replacement (requires `nprox > 0`)

Both of the above options require setting
`code = missing value code`

Note: the out-of-bag error rate is biased when
`missfill=2`

Graphics

requires:

- `iviz = 1`
- `impn=1`
- `imp=1`
- `nscale=3`
- `nprox=1`

optionally: `nprot=p` and `int=1` give additional plots in the display

4. Special purpose settings

- `ndsize` = number of cases in terminal node
- `look` = prints intermediate results
- `lookcls` = 1 prints individual class results
- `interact` = 1 computes variable interactions
- `impn` = casewise variable importance
(requires `imp` = 1)
- `impfast` = 1 fast variable importance (does not require `imp`=1)

Special purpose settings (continued)

- `mselect = 1` allows a subset of variables to be used throughout

Go to line 265:

```
c SELECT SUBSET OF VARIABLES TO USE
```

```
if(mselect.eq.0) then
```

```
    mdimt=mdim
```

```
    do k=1,mdim
```

```
        msm(k)=k
```

```
    enddo
```

```
endif
```

- `iseed = 4351` seed for the random number generator
- `mfixrep` = the number of iterations for careful replacement of missing values

Special purpose settings (continued)

- `isaverf`, `irunrf` to save forest and re-run
- `isavepar`, `ireadpar` to save parameter settings and recall
- `isavefill`, `ireadfill` to save missing value fills and re-use
- `isaveprox`, `ireadprox` to save proximities and re-use

5. Output control

Around line 70:

```
parameter(  
&      isumout= 1,    !0/1    1=summary to screen  
&      idataout= 1,   !0/1/2  1=train,2=adds test (7)  
&      impfastout= 1, !0/1    1=gini fastimp (8)  
&      impout= 1,     !0/1/2  1=imp,2=to screen (9)  
&      impnout= 1,    !0/1    1=impn (10)  
&      interout= 1,   !0/1/2  1=interaction,2=screen (11)  
&      iprotout= 1,   !0/1/2  1=prototypes,2=screen (12)  
&      iproxout= 1,   !0/1/2  1=prox,2=adds test (13)  
&      iscaleout= 1,  !0/1    1=scaling coords (14)  
&      ioutlierout= 1) !0/1/2  1=train,2=adds test (15)
```