



THE
POWER
TO KNOW.

**SAS[®] Enterprise Miner[™] and
SAS[®] Text Miner Procedures
Reference for SAS[®] 9.1.3
The PMBR Procedure
(Book Excerpt)**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® *Enterprise Miner™* and SAS® *Text Miner Procedures: Reference for SAS® 9.1.3*, Cary, NC: SAS Institute Inc.

SAS® Enterprise Miner™ and SAS® Text Miner Procedures: Reference for SAS 9.1.3

Copyright © 2008 by SAS Institute Inc., Cary, NC, USA.

All rights reserved. Produced in the United States of America.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

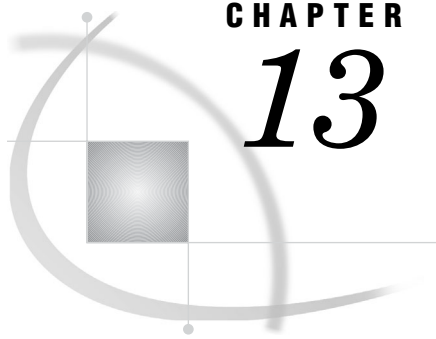
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, October 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at support.sas.com/publishing or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.



CHAPTER

13

The PMBR Procedure

<i>Overview: PMBR Procedure</i>	317
<i>Syntax: PMBR Procedure</i>	318
<i>PROC PMBR Statement</i>	318
<i>VAR Statement</i>	320
<i>TARGET Statement</i>	321
<i>CLASS Statement</i>	321
<i>ID Statement</i>	321
<i>SCORE Statement</i>	321
<i>DECISION Statement</i>	322

Overview: PMBR Procedure

The PMBR procedure is used for prediction as an alternative to other predictive modeling techniques in Enterprise Miner, such as the NEURAL, SPLIT, DMSPLIT, DMNEURL, and DMREG procedures. However, the technique in the PMBR procedure is different. Whereas all the other techniques attempt to determine some rules for predicting future examples, the PMBR procedure categorizes an observation in a score data set by retrieving its *k* closest neighbors from a training data set, and then having each neighbor vote on the target value based on its value for the target variable. These votes then become the posterior probabilities for predicting the target, which are included in an output data set. Training is thus faster than with the alternative techniques, but scoring is generally slower.

The target variable is expected to be either binary, interval, or nominal. Ordinal targets are not specially supported at this time, but could be modeled as interval targets. If the target variable is a class variable in the DMDB, one variable is created on the output data set for each value of the target, representing the appropriate posterior probabilities. Otherwise, one predicted variable is created on the output data set corresponding to the average prediction for the *k* neighbors.

The neighbors are determined by a simple Euclidean distance between the values on each of the variables in the VAR statement for the probe and target example. Thus, it is assumed that the variables are orthogonal to each other and standardized. If your input data is not in that form, you need precede this procedure with one that will create numeric, orthogonal, and standardized variables — such as the PRINCOMP, DMNEURL, PRINQUAL, CORRESP, SPSVD procedures.

The PMBR procedure needs to be run separately and be given the DMDB-name for each of the data sets to be scored, including any training, validation, test, or score data set.

Missing values in either the training or score data set are replaced by the mean of that variable as stored in the DMDB catalog.

Syntax: PMBR Procedure

PROC PMBR

```

DMDBCAT= <libref.>SAS-catalog>
<BUCKETS= positive number>

<DATA=<libref.>SAS-data-set>
<EPSILON= positive number>
<K=integer>
<METHOD=method>
<NEIGHBORS>
<OPTIMIZEK>
<OUT= <libref.>SAS-data-set>
<OUTEST= <libref.>SAS-data-set>
<SCORE=<libref.>SAS-data-set>
<SHOWNODES>
<TESTDATA= <libref.>SAS-data-set>
<THREADS | NOTTHREADS | THREADS=positive integer>
<VALIDATA= <libref.>SAS-data-set>
<WEIGHTED>;
<VARvariable>;
TARGET <variable>;
<CLASS variable>;
<IDvariable(s)>;
<SCORE>
  <DATA=<libref.>SAS-data-set>
  <OUT= <libref.>SAS-data-set>
  <OUTFIT= <libref.>SAS-data-set>
  <ROLE= TRAIN, VALID, TEST, SCORE>;
<DECISION>
  <COSTVARS= variable(s) or number(s)>
  <DECDATA=<libref.>SAS-data-set>
  <DECVARS= variable(s)>
  <PRIORVAR= variable>;

```

PROC PMBR Statement

Invokes the PMBR procedure.

```
PROC PMBR <option(s)>;
```

Required Arguments

```
DMDBCAT = <libref.>SAS-catalog
```

Specifies the DMDB catalog.

Options

BUCKET = *positive integer*

Indicates the number of buckets to allow a leaf node to grow before splitting into a branch with two new leaves. This value must be greater than or equal to 2. This option only applies to the KD-Tree or RD-Tree methods.

Default: 8

DATA = (or IN =) <libref.>SAS-data-set

Specifies the input SAS data set to be trained on. If you omit the DATA= option, the procedure uses the most recently created SAS data set.

EPSILON = *positive number*

Indicates an approximate nearest neighbor search when a non-zero number is specified, where the nearest neighbors determined so far must be at most “epsilon” away from the actual nearest neighbors to terminate the search. For large dimensionality, judicious use of epsilon can result in radically improved performance. This option only applies to the KD-Tree or RD-Tree methods.

Default: 0.0

K = *integer*

Specifies the number of nearest neighbors to retrieve.

Default: 1

METHOD = *method*

Determines what data representation is used to store the training data set and then to retrieve the nearest neighbors. The following methods are available:

SCAN	Retrieves a nearest neighbor by naively going through every observation in the training data set and calculating its distance to a probe observation.
RDTREE	Uses an RD-Tree to store the observations in the training data set in memory. This is a proprietary representation that, like a KD-Tree, also operates in $o(\log n)$ time, but will generally examine fewer nodes than an RD-Tree to find the neighbors, and can be applied with somewhat greater dimensionality.

NEIGHBORS

This option specifies to write the nearest neighbors for each observation in the SCORE data set to the OUT data set. The SAS variables in the OUT data set denoting the nearest neighbors are `_N1`, `_N2`, ..., `_NK`, where K is the number of nearest neighbors specified. The values of the variables `_Ni` are the values of the ID variable (if ID variable was specified) or positive integers corresponding to row number in the DATA= data set.

Note: If an ID variable was not specified, then the values of the nearest neighbor variables `_Ni` are no longer valid if the DATA= data set gets sorted. △

OPTIMIZEK

This option will find the optimal value of K (the number of nearest neighbors) between 1 and K. For an interval target, the optimal value of K is the smallest number of nearest neighbors that corresponds to the largest correlation between the actual target variable and the predicted target variable in the training data set. For a categorical target, the optimal value of K is the smallest number of nearest

neighbors that corresponds to the largest sum of posterior probabilities of the level of the target variable in the training data set.

OUT = <libref.>SAS-data-set

Specifies the name of the output data set. This output data set contains all variable in the score data set and additional variables representing the posterior probabilities. If the target variable is categorical, the names of these variables generally begin with P_, followed by a part of the original variable names and with the values added to the end. These posterior probabilities correspond to the percentages of the k neighbors that have the value as the target. If the target variable is interval, a single posterior variable is produced that averages the target values across the k neighbors. If the OUT=option is not specified when the SCORE=option is used, DATA1,...,DATA_n will be created automatically.

OUTEST= <libref.>SAS-data-set

This option specifies the name of the data set containing the estimated weights and the fit statistics for the training, validation, and test data sets.

SCORE = <libref.>SAS-data-set

Specifies the data set to be scored. This data set might not have the target variable. It can be the same name as the training data set.

SHOWNODES

Includes a variable `_nnodes_` in the output data set that shows the number of point comparisons that had to be done to determine the answer. This is useful as a point of comparison.

TESTDATA= <libref.>SAS-data-set

This option specifies the name of the test data set. The validation data set should be raw data, not a DMDB encoded data set. Also, this data set must contain the input variables and the target variable as in the DATA= data set.

THREADS | NOTTHREADS | THREADS= <n>

The THREADS | NOTTHREADS option enables or disables the activation of a multi-threaded nearest neighbor search. Setting THREADS=*n*, where *n* is an integer greater than 1, will force *n* threads to be spawned. The option METHOD=RDTREE is not supported with multiple threads.

Default: The default is THREADS.

VALIDATA= <libref.>SAS-data-set

This option specifies the name of the validation data set. The validation data set should be raw data, not a DMDB encoded data set. Also, this data set must contain the input variables and the target variable as in the DATA= data set.

WEIGHTED

Specify this option if you want the dimensions weighted according to the absolute value of their correlation with the target variable. This option is ignored (and a note printed) if the target is a class variable with more than two levels.

VAR Statement

VAR <variable>;

variable

Specifies all numeric variables that you want to treat as dimensions for the nearest neighbor lookup. These should be standardized and orthogonal for the nearest neighbor search to be accurate. If no VAR statement is specified, all numeric variables in the data set will be used.

TARGET Statement

Specifies one variable to be used as the target. It can be numeric or character. This statement is required.

TARGET <variable>;

CLASS Statement

This is currently ignored. The procedure determines whether the target is a class variable based on the contents of the DMDB, and it cannot be changed in this procedure.

CLASS <variable>;

ID Statement

The ID statement should specify one numeric variable to be used as an identifier for each observation. No two observations in the DATA=data set should have the same value for the ID variable.

ID <variable(s)>;

SCORE Statement

SCORE <option(s)>;

Options

DATA = <libref.>SAS-data-set

Specifies the name of the data set to score.

OUT= *<libref.>SAS-data-set*

Specifies the name of the output data set containing predicted values, posterior probabilities, residuals, and so on.

Default: The default name is DATA n , where n is a positive integer.

OUTFIT= *<libref.>SAS-data-set*

Specifies the name of the data set containing fit statistics for the data set to score.

ROLE= *TRAIN, VALID, TEST, SCORE*

Specifies the type of fit statistics computed for the OUTFIT= data set.

ROLE=SCORE is not allowed when OUTFIT= is specified. If ROLE=TRAIN is specified, then the data set to score must be the same as the training data set.

DECISION Statement

DECISION *<option(s)>*;

Options

COSTVARS (COSTVAR or COST)=*variable(s) or number(s)*

This list specifies the cost variables and/or cost constants. The cost variables must appear in the data set to score.

DECDATA (or DECISIONDATA)= *<libref.>SAS-data-set*

Specifies the name of the data set containing the decision matrix and prior probabilities. This data set must be specified.

DECVARs (or DECVAR)=*variable(s)*

The decision variables in the DECDATA= data set.

PRIORVAR=*variable*

The variable in the DECDATA= data set that represents the prior probabilities for a categorical target.