Paper 081-2007

Variable Screening for Multinomial Logistic Regression on Very Large Data Sets as Applied to Direct Response Modeling

John A. Cherrie, Alliant Cooperative Data Solutions, Brewster, NY

ABSTRACT

Beginning with version 8.2 SAS supports multinomial logistic regression as part of PROC LOGISTIC. As a direct response modeling provider, we have found that multinomial logistic regression models often provide us with our best solutions.

However, using multinomial logistic regression presents some challenges. We are often faced with very large sample sizes and a large number of candidate predictor variables. The sheer sizes of our data sets make variable selection and scoring significant issues. The functionality that SAS has built into PROC LOGISTIC to handle variable selection and scoring is not feasible for the size of data sets that we use. I will present how we have addressed the issue of variable screening in our modeling environment.

I will provide a comparison of gains charts that show that the multinomial logistic model is preferable to other solutions. I will discuss and share bits of code on how we utilize univariate screening, resampling methods, and clustering for variable reduction.

Discussing the issues regarding creating efficient production scoring modules utilizing multinomial logistic regression is beyond the scope this presentation. Upon request I will supply code for a simple macro that creates a text file version of a coefficient file that can be incorporated into an efficient production scoring environment.

INTRODUCTION

A classic problem that challenges direct response regression modelers is that high responding names tend to be poor paying names. Most direct response modelers build separate models for response and for payment. These separate models are then combined in a variety of ways. One popular method is to create a cross-tabulation of model *vigintiles/vigsimals*[1] that is often called a *matrix model*. Another popular method is to weight the independently estimated probabilities with profit values so that the result is a single value of estimated profitability. The second type of model is often called an *expect profit value* or *expected net present value* model.

Regardless of how separate models are combined, the model that identifies gross responders tends to be negatively correlated with the model that identifies payment propensity. The result is that customers with a high probability of response have a low probability of actually paying for the product.

Beginning with version 8.2 SAS supports multinomial logistic regression as part of PROC LOGISTIC[2]. At Alliant Cooperative Data Solutions we have found that multinomial logistic regression models successfully segment customers that are both high responding and high paying. We have had phenomenal success using probabilities estimated from multinomial logistic regression models to calculate expected profit value models. The gains charts that are estimated in this fashion tend to provide a monotonic ranking with respect to profitability and net response and a *u-shaped* distribution with respect to gross response. Thus, we are able to identify customers with a high probability of responding and a high probability of paying.

However, building a multinomial logistic regression model in SAS presents a variety of challenges for direct response modelers. Direct response modelers typically have enormous size data sets. A typical data set might have a few hundred thousand names and thousands of candidate variables.

Model Interpretation, variable selection, and scoring which are straightforward issues for binomial logistic regression become complex issues for multinomial logistic regression. SAS has addressed the issues of variable selection (with the SELECTION=STEPWISE option in the MODEL statement) and scoring (with the SCORE statement).

---

[1] A vigintile or vigesimal is a grouping of names consisting of 5% of the total distribution.

[2] SAS has supported generalized logit modeling via PROC CATMOD since version 6. However, using PROC CATMOD is best suited for a relatively small number of categorical predictors. It is awkward for fitting models with many continuous predictors and the resulting estimation file is difficult to utilize.

Unfortunately, given the enormous size of data sets that are typically used in a direct response environment, stepwise selection is unsuitably slow for multinomial variable selection. During this presentation I will to present how we solved these challenge of variable selection. I will provide an overview of the process and briefly illustrate the macro code we developed. Because of time constraints, I will not be presenting how we customized scoring multinomial logistic regression models.

CASE STUDY: COMPARISON OF SIMPLE LOGISTIC DIRECT RESPONSE MODEL VERSUS EXPECTED PROFIT VALUE MODEL USING GENERALIZE LOGISTIC MODEL

Applying generalized logistic regression to very large data sets presents challenges. There needs to be significant benefit to justify the additional work. The following case study illustrates why using the generalized logistic regression model is often worthwhile.

The objective for this client was to identify the least profitable customers to save on promotion cost.

SIMPLE LOGISTIC MODEL APPLIED TO DIRECT RESPONSE CLIENT

The following table represents keys statistics from a simple 'net paid' logistic regression model developed for a direct response client.

	Mail Volume	Response		Payment>=$25.00			Profit Estimate	
Group	Cum %	%	Index	Net Rate	Net Index	% of Orders	Per Mail Piece	Per Customer Paid>=$25
1	5.0%	4.6%	294	2.7%	266	58.6%	$0.70	$26.40
2	10.1%	3.1%	202	1.9%	194	62.2%	$0.49	$25.79
3	15.3%	2.5%	163	1.6%	162	64.4%	$0.41	$25.59
4	20.1%	2.2%	143	1.5%	146	66.1%	$0.43	$26.24
5	25.1%	2.0%	126	1.3%	133	68.1%	$0.27	$25.32
6	30.1%	1.8%	115	1.1%	113	64.0%	$0.19	$24.32
7	35.1%	1.7%	109	1.1%	113	67.5%	$0.20	$23.62
8	40.2%	1.5%	100	1.0%	103	66.8%	$0.20	$23.28
9	45.2%	1.5%	95	0.9%	92	62.6%	$0.17	$22.91
10	50.1%	1.4%	91	0.9%	92	65.2%	$0.20	$22.86
11	55.2%	1.3%	83	0.9%	91	70.8%	$0.24	$23.06
12	60.1%	1.3%	83	0.8%	84	65.8%	$0.12	$22.60
13	65.2%	1.0%	63	0.6%	60	61.4%	$0.05	$22.09
14	70.2%	1.0%	64	0.7%	69	69.1%	$0.10	$21.75
15	75.1%	1.0%	62	0.7%	68	71.0%	$0.07	$21.32
16	80.1%	0.8%	52	0.5%	52	65.3%	$0.01	$20.78
17	85.0%	0.8%	51	0.5%	54	69.5%	$0.05	$20.48
18	90.0%	0.7%	47	0.5%	45	62.1%	($0.01)	$19.96
19	95.0%	0.5%	30	0.3%	35	73.9%	($0.06)	$19.31
20	100.0%	0.3%	21	0.3%	25	77.3%	($0.13)	$18.41
Totals	100.0%	1.6%	100	1.0%	100	64.8%	$0.19	$18.41

The table above shows that the simple logistic regression model did what it was supposed to. However, notice that the percentage of orders with payment that is greater than $25.00 is the largest at the bottom of the gains chart!

The model above does a very adequate job of ranking response and net payments above $25.00. However, it does a poor job of ranking payment rates. The most important aspect of a direct response model is its ability to rank profit. Only the bottom 3 groups (18, 19, and 20) show a negative profit.

EXPECTED PROFIT VALUE MODEL APPLIED TO THE SAME DIRECT RESPONSE CLIENT

Since the simple logistic regression model did only a marginal job of ranking the most profitable customers, the model was redeveloped using an Expected Profit Value model or EPV model. The EPV model used generalized logistic regression using PROC LOGISTIC to estimate the probabilities of various customer behaviors.

	Mail Volume	Response		Payment>=$25.00			Profit Estimate	
Group	Cum %	%	Index	Net Rate	Net Index	% of Orders	Per Mail Piece	Per Customer Paid>=$25
1	5.1%	3.3%	214	2.6%	257	77.8%	$0.91	$35.31
2	10.1%	2.5%	158	1.8%	183	74.9%	$0.68	$36.10
3	15.2%	2.1%	135	1.6%	155	74.1%	$0.38	$33.11
4	20.2%	1.8%	117	1.4%	136	75.7%	$0.31	$31.17
5	25.3%	2.1%	135	1.4%	143	68.3%	$0.34	$29.99
6	30.2%	1.8%	115	1.2%	122	68.7%	$0.35	$29.84
7	35.1%	1.5%	94	1.0%	100	68.6%	$0.17	$28.70
8	40.0%	1.6%	100	1.0%	103	66.6%	$0.24	$28.22
9	45.4%	1.6%	104	0.9%	90	56.0%	$0.09	$26.86
10	50.0%	1.4%	88	0.9%	90	66.9%	$0.22	$26.68
11	55.0%	1.4%	91	0.9%	94	67.0%	$0.17	$26.16
12	60.1%	1.1%	73	0.8%	77	69.2%	$0.09	$25.42
13	65.7%	1.1%	72	0.7%	66	59.7%	$0.06	$24.65
14	70.1%	1.0%	64	0.7%	65	65.5%	$0.09	$24.26
15	75.0%	1.0%	67	0.7%	66	64.0%	$0.06	$23.69
16	80.0%	0.9%	58	0.6%	58	65.2%	($0.05)	$22.65
17	85.1%	0.8%	51	0.4%	44	56.1%	($0.05)	$21.81
18	90.1%	0.9%	60	0.5%	45	48.6%	($0.10)	$20.76
19	95.1%	0.8%	50	0.4%	43	55.8%	($0.08)	$19.86
20	100.0%	2.3%	150	0.6%	60	25.8%	($0.18)	$18.41
Totals	100.0%	1.6%	100	1.0%	100	64.8%	$0.19	$18.41

The table above shows that the EPV model that utilized the generalized logistic regression model ranks customers quite differently. Group 20 actually exhibits a fairly high response rate of 2.3%. However, this group also has a the smallest percentage of customers with payment greater than $25.00. Group 20 is also very unprofitable.

Overall, many direct response clients would find this model more appealing. It provides greater profitability at a lower mail volume. This model also does a better job of controlling back end performance.

The form of the EPV model is:

$$EPV = \sum_{\text{all customer actions}} \text{probabilty(customer action)} * \text{value(customer action)}$$

```
epv=sum(prob1*-0.30,
        prob2*-12.3140503,
        prob3*-20.8204920,
        prob4*16.6652950,
        prob5*141.8555110);

label   prob1='p_nonresp coded 0'
        prob2='p_unpaid coded 10'
        prob3='p_return coded 20'
        prob4='p_pay_cancel coded 30'
        prob5='p_paid coded 40';
```

The probabilities for the EPV model were estimated using PROC LOGISTIC with the LINK=GLOGIT option::

```
PROC LOGISTIC DATA=client_x_development REF=first SIMPLE OUTEST=cf_client_x_glogit
              NAMELEN=32;
     MODEL multivar=predictor_variable1--predictor_variable15/LINK=GLOGIT;
     TITLE "Client X Generalized Logistic Regression Model";
     WEIGHT WT_VAR/normalize;
     SCORE data=lip.client_x_validation out=client_x_validation_scored;
     SCORE data=lip.client_x_development out=client_x_development_scored;

RUN;
```

VARIABLE SCREENING ON VERY LARGE DATA SETS

Direct response modelers typically work with very large data sets. Typically, one might develop a model on a sample with 1,000 candidate predictor variables and perhaps 100,000 records.

USING SELECTION=STEPWISE WITH LINK=GLOGIT

One possibility to screen variables is to use SELECTION=STEPWISE with LINK=GLOGIT. Unless you are working with a very small data set I do not recommend this for selecting variables.

Here are some benchmark numbers for a data set with 30 candidate variables and 300,541 records:

```
NOTE: There were 300541 observations read from the data set
WORK.MATCH_DEV_TRANSFORMED.
NOTE: The PROCEDURE LOGISTIC printed pages 1-36.
NOTE: PROCEDURE LOGISTIC used (Total process time):
      real time           3:45:58.51
      cpu time            3:41:17.70
```

Consider how that compares to a standard stepwise logistic regression on the same data:

```
NOTE: There were 300541 observations read from the data set
LIP.LIP_MATCH_DEV_TRANSFORMED.
NOTE: The PROCEDURE LOGISTIC printed pages 1-39.
NOTE: PROCEDURE LOGISTIC used (Total process time):
      real time           10:09.93
      cpu time            9:13.25
```

The stepwise generalized logistic regression took over 20 times longer than a standard stepwise selection!

A typical data set for us will have 4,000 or more candidate predictor variables and 500,000 observations. 5 bytes of precision is usually enough for our numeric data but that still yields a data set of approximately 10 GB! Even with very powerful machines stepwise selection of variables of this size of a data set will run for days and still not finish!

CUSTOM VARIABLE SELECTION MACRO

An alternative was clearly needed. Thus, we have created an automated four step approach to evaluating the potential contribution of each candidate variable to a final multinomial model. We put all the candidate variables in a SAS data set called a variable table. Each candidate variable has a row in the table. The columns represent various statistics for each step in the selection process. The variable table is easy to produce using output from SAS ODS and shows how each variable fared during every step of our variable screening process.

The inputs to the process are as follows: Data Set, Weight Variable (optional), P-value (for significance testing), and a Dependent Variable.

THE OUTPUT OF THE PROCESS IS A REDUCED DATA SET THAT CONTAINS ONLY THOSE VARIABLES THAT SURVIVED THE SCREENING. IN ADDITION, WE CREATE Physical Size:416.6 MB
a SAS data set called a *variable table* that contains statistics from each step in the process.

The initial variable table contains nothing more than the name of the candidate variable as a field. As each step of the macro executes the variable table adds the relevant statistics from each step. At the end of the process, the modeler can use the variable table to adjust the criteria to add or reduce the final number of variables selected.

The steps involved in our variable screening process are as follows:

1.) Elimination of trivial candidate variables
2.) Simple ANOVA 1 variable at a time
3.) Bagging (resampling) simple logistic stepwise models
4.) Variable clustering
5.) Output Screened data set

Each step is actually performed via 4 separate macros (except that steps 1 and 2 are combined into a single macro). Each of the 4 macros is executed from an overall *driver* macro. The system was designed in this way so that each step could be run separately or as an entire process.

STEP 1: ELIMINATION OF TRIVIAL CANDIDATE VARIABLES

Candidate variables that have a standard deviation of 0 can be eliminated with no further evaluation. We also eliminate variables with a very small standard deviation. Practical experience based on the range of the data has to dictate what levels to eliminate. For example, if a candidate variable is a 0, 1 dummy variable, variables with a mean of less 0.0001 will almost never add any practical value to a model. Variables that pass this screen become candidates for simple ANOVA.

The following is a sample of code that is used within the first macro:

```
proc means noprint data=temp;
%if %bquote(&weight) ne %then %str(weight &weight;);
output out=all_means;
run;

proc transpose data=all_means(drop=_type_ _freq_ )
                out=all_means_trans(drop=_label_ rename=(_name_=variable));
id _stat_;
run;

*** eliminate records with no variation - eliminate dummy/continuous variables with
less than 0.01% mean;

proc sql;
 create table &inputds._keeplist as
 select *
 from all_means_trans
 where (std ne 0) and not(min=0 and mean<&meanmin) and (n>0);
 run;
```

We do not preserve trivial variables on the variable table. They are not useful and there is not need to maintain them on the final variable table.

STEP 2: SIMPLE ANOVA ONE CANDIDATE AT A TIME

Hosmer and Lemeshow[3] discuss the univariate selection of variables for logistic regression using t-tests. They suggest using a p-value of 0.25 to screen independent variables. Because of the shear number of predictors that are available in the direct response environment, practical experience has shown the p-values as small as 0.05 will not lead to a significant increase in Type II errors. Our approach has been to perform an ANOVA using the response levels as the dependent variable and the candidate variable as the predictor variable.

Practical experience has shown that standard ANOVA produces a variable set that is very similar to non-parametric ANOVA. Missing values are ignored. Variables that pass this screen become candidates for bagging.

[3] Hosmer DW, Lemeshow S. Applied Logistic Regression. New York: John Wiley & Sons; 1989.

The following is a sample of code within the first macro that shows how the macro calculates ANOVA estimates for each independent variable. In this case, the continuous *predictor* variable acts as the independent variable.

```
*------------------------------------------------------------*
|  Loop through all Numeric Variables in Data Set            |
*------------------------------------------------------------*;

%do range=1 %to &numobs %by 200;
%let upper=%eval(&range+199);

  data varlist;
     set &inputds._keeplist;
        if &range<=_n_<=&upper ;
  run;

 data _null_;
  set varlist NOBS=numobs end=last;
   call symput(cats("var",_n_),variable);
   if last then call symput("numobs",numobs);
 run;

 proc corr data=&inputds. OUTP=%cmpres(corr_&range._&upper) NOPRINT;
 var  %do j=1 %to &numobs; &&var&j %end; &dumlist;
     %if %bquote(&weight) ne %then %str(weight &weight;);
 RUN;

%put Output Correlation Data Set Will be Called: %cmpres(corr_&range._&upper);

ods listing close;
 data _null_;
  set varlist   end=last;
   call execute("ods output Test ANOVA=reg_temp(KEEP=MODEL SOURCE DF MS FVALUE PROBF
WHERE=(Source='Numerator'));");
   call execute("proc REG data=%cmpres(corr_&range._&upper) ;");
   call execute(variable||": model "||variable||"=&dumlist            ;");
   call execute("test &dumlist_comma;");
   call execute("RUN;");
   call execute("quit;");
   call execute("ods output close;");

   IF EXIST("WORK.REG_TEMP") THEN DO;
   call execute("Proc APPEND base=%cmpres(work.reg_total_&out_suffix)
data=reg_temp(drop=source) force;");
   call execute("run;");
   END;
RUN;
%end;
```

Notice that the routine loops through sets of 200 variables at a time. A correlation matrix is formed once and individual regressions are run using the correlation matrix. PROC REG can run produce ANOVA statistics for 200 variables at a time using the correlation matrix as input.

variable	DF	MS	FValue	ProbF
AD_MAG_D90_ORD_AMT	1	40760	11.14	0.0008
AD_MAG_D90_PAY_AMT_CASH	1	24482	10.98	0.0009
AD_MAG_D90_PAY_AMT_TOTAL	1	30806	10.45	0.0012
AD_MAG_D90_PAY_COUNT_CASH	1	43.65182	8.97	0.0027
AD_MAG_D90_PAY_COUNT_TOTAL	1	53.86412	7.92	0.0049
AD_MAG_TOT_BILL_LVL_INVOICE	1	5073.62066	7.68	0.0056
AD_MAG_TOT_ORD_COUNT	1	20017	9.73	0.0018
AD_MAG_TOT_PAY_AMT_CASH	1	5651905	16.35	<.0001

STEP 3: BAGGING

The third step in our variable selection process is to perform a bagging run is then preformed on the survivors of ANOVA. Bagging is another name for Bootstrap Aggregation. Bootstrap replicates of a data set are created and models are fit to each replicate. The predictions are averaged for each model. Proponents indicate that it stabilizes predictions. The theory is not fully explained but we have found it useful as a component in variable selection.

For the survivors of step 2, large groups of variables (thousands) are shuffled and dealt into groups of 20 to 40 candidate variables. Large numbers of observations are sampled down from between 20,000 to 40,000 records. A stepwise logistic regression is run for all variable groups, and with these dimensions, the stepwise will run fast. This process is repeated 10 times. Common results are reinforced, and oddball occurrences are isolated.

Missing value flags are created and forced into each bagging run. Bagging runs are performed on each level of the dependent variable. Variables survive this step if they are selected for at least one the dependent variables.

```
data calc_sample ;
  depvar_0_input = &_nobs_depvar_0 ;
  depvar_1_input = &_nobs_depvar_1 ;
  min_resprate   = &min_resprate. ;
  max_regsamp    = &max_regsamp.  ;
  totsamp_input = depvar_0_input + depvar_1_input ;
  resprate_input = depvar_1_input / totsamp_input;
  regsamp_available = min( depvar_0_input + depvar_1_input, max_regsamp) ;
  resprate_avail = depvar_1_input / regsamp_available  ;
  if resprate_avail ge min_resprate then do ;
       path = 'top';
       resprate_used = max(resprate_input, min_resprate) ;
       sampresp_count = min(ceil(resprate_used*regsamp_available), depvar_1_input) ;
       sampnrsp_count = min (regsamp_available - sampresp_count, depvar_0_input);
  end ;
  else do ;
       path = 'bot';
       sampresp_count = depvar_1_input ;
       sampmax_based_on_depvar_1 = ceil( depvar_1_input / min_resprate ) ;
       sampnrsp_count=min(sampmax_based_on_depvar_1-sampresp_count, depvar_0_input);
  end;

  totsamp_output_count = sampresp_count + sampnrsp_count ;
  resprate_output_count = sampresp_count / totsamp_output_count;
 sampresp_rate=min(ceil(totsamp_input*sampresp_count/depvar_1_input),totsamp_input);
 sampnrsp_rate=min(ceil(totsamp_input*sampnrsp_count/depvar_0_input),totsamp_input);
    call symput ('resprate_input',left(put(resprate_input,6.4))) ;
    call symput ('sampresp',left(put(sampresp_rate,9.))) ;
    call symput ('sampnrsp',left(put(sampnrsp_rate,9.))) ;
    call symput ('totsamp',left(put(totsamp_output_count,9.))) ;
    call symput ('sampresp_count',left(put(sampresp_count,9.))) ;
    call symput ('sampnrsp_count',left(put(sampnrsp_count,9.))) ;
    call symput ('totsamp_output_count',left(put(totsamp_output_count,9.))) ;
    call symput ('resprate_output_count',left(put(resprate_output_count,6.4))) ;
 output;
run;

%numobs(data=varin,macrovar=varin);

%do iloop = 1 %to &iter   ;
%if  &numvgrps > 1  %then %do ;
data varin;
  set varin (keep = name)  ;
   randno = ranuni( 3 * &iloop * &randseed );
run;
proc sort data = varin;
  by randno;
run;
%end;
```

```
data _null_  ;
  set varin ;
  call symput ('vn'!!left(put(_n_,6.)),name) ;
run;

data ptrresp (keep = ptr sequence_id );
    ptr = 0 ;
  do recno = 1 to &sampresp;
  ptr = 1 + int( ranuni(7 * &iloop * &randseed) * &totfile) ;
  point=ptr;
   set depvar point = point;
    if depvar = 1 then output;
  end;
  stop;
run;

data ptrnrsp (keep = ptr sequence_id );
 ptr = 0 ;
  do recno = 1 to &sampnrsp;
  ptr   = 1 + int( ranuni(9 * &iloop * &randseed) * &totfile) ;
  point=ptr;
   set depvar point = point   ;
    if depvar = 0 then output;
  end;
  stop;
run;

data allpoint;
  set ptrresp ptrnrsp;
run;
proc sort data = allpoint;
   by sequence_id ;
run;

  *  take sample using merge by sequence_id ;
  data samp   ;
   merge allpoint (in = a)
     sdlana_temp (in = b keep =sequence_id &depvar &weight &missing_list
       %do v=1 %to &infobs; &&vn&v %end; );
       by sequence_id ;
       if a and b;
   run;


%do n= 1 %to &numvgrps ;
  %let itergrp = I&&iloop.G&n ;
  %put Up To Iteration Group &itergrp ;
  %let fobs = %eval(1+ (&n-1)* &vclump) ;
  %let lobs = %eval(&n * &vclump ) ;
  %if  &lobs > &infobs %then %let lobs=&infobs ;

 PROC LOGISTIC DESCENDING OUTEST=OE
   %if &&iloop > 1 %then %do; noprint %end;  DATA = SAMP ;
  %if (&weight gt '') %then %do ;
 %if (%upcase(&normweight) = Y) %then %do ;
  weight &weight / norm ;
 %end ;
 %else %do ;
  weight &weight  ;
 %end ;
 %end;

 MODEL &DEPVAR =&missing_list
  %do v=&fobs %to &lobs; &&vn&v %end;
  &modelopt INCLUDE=&missing_count; ;
 run;
```

```
 PROC TRANSPOSE DATA = OE OUT=OETrun; run;

 data oetrun;
 length _name_ $ 32 ;
 set oetrun;
 *if (_name_ = '_LNLIKE_') or
    (ESTIMATE=.) then delete;
 run;

 %if &itergrp= I1G1  %then %do;
     DATA OET; set oetrun;
     run;
 %end;
 %else %do;
     PROC APPEND BASE =  OET NEW = oetrun FORCE;
     run;
 %end;
 %end; %* of numvgrps loop;
 %end ; %* end of master loop for iterations ;

proc sort data =oet (where = (&depvar. ne .) rename = (_name_ = varname))
          out = oetsorted ;
     by varname;
run;
data result (keep = varname selected_count selected_pct);
 set oetsorted; where index(upcase(varname),'X_IND')=0;** exclude missing flags **;
  by varname;  drop _label_;
  selected_count +1 ;
  if last.varname;
  selected_pct = selected_count / &iter.;
  output; selected_count = 0 ;
run;
```

The bagging output is as follows:

VAR	PCTSEL	SELCNT	ESTMEAN	POSITEST	STD ERR	T
GS_CONT_TOT_COUNT_PER_TB_HH	100	4	0.42	1	0.03	12.35
GS_CONT_TOT_COUNT_PER_TB_IN	100	4	0.60	1	0.05	12.37
GS_PAY_CC_COUNT_PER_TB_HH	100	4	-0.96	0	0.11	8.87
GS_PAY_CC_COUNT_PER_TB_IN	100	4	-1.23	0	0.14	9.00
GS_WO_AMT_LAST_YR_PER_TB_HH	100	4	0.06	1	0.01	8.21
HH_BK_TOT_BILL_LVL_WO_RATE	100	4	-0.38	0	0.05	7.86

STEP 4: VARIABLE CLUSTERING

At this stage of variable selection we would have greatly reduced the number of surviving candidate variables. However, it is still likely that highly candidate correlated variables still exist. We have found that an automated step using PROC VARCLUS to find a single variable that represents each variable cluster greatly reduces the candidate set with no loss of predictiveness.

PROC VARCLUS attempts to divide a set of variables into non-overlapping categories (clusters). The goal is to find groups of variables that are as correlated as possible among themselves and as uncorrelated as possible with variables in other clusters. Large set of variables can be replaced by a single member of each cluster to act as a representative with very little loss of information

The VARCLUS procedure is closely related to principal components analysis. The basic algorithm is binary and divisive. All variables start in one cluster. A principal component analysis is done on all the variables in the cluster. If the second eigenvalue is greater than a specified threshold then the cluster is split. The PC scores are then rotated so that the variables can be split into two groups. This process is repeated for the two child clusters until the second eigenvalue drops below the threshold.

In order to choose the *best* variable in a cluster, we chose to use the 1-Rsquare ratio as the criterion. Small values

of this ratio indicate that the variable has a strong correlation with its own cluster and a weak correlation with the other clusters.

The following is a sample of the variable clustering code:

```
ods listing close;
 ods output clusterquality=summary rsquare(match_all)=clusters;

 proc varclus data=&DATASET HI MAXEIGEN=0.7 /*MAXC=10*/ outstat=vclus_out1
      %if %bquote(&maxclusters) ne %then %do; maxclusters=&maxclusters %end;;
  var &varlist;
  %if %bquote(&wt) ne %then %do; weight &wt; %end;
  run;

 data _null_;
  set summary;
  call symput('ncl1',trim(left(numberofclusters-2)));
  call symput('ncl2',trim(left(numberofclusters)));
 run;

 data vclus_out2;
  length _name_ $30.;
  set vclus_out1;
  if _ncl_=&ncl2 and _type_='GROUP';
  drop _ncl_ _type_ _name_;
 run;
 proc transpose data=vclus_out2 out=vclus_out3 (rename=(col1=Cluster name_=Variable
                 _label_=VarLabel));
 run;

 data scores;
  set vclus_out1;
  where _ncl_=&ncl2 and _type_='SCORE';
  drop _ncl_ _type_;
 run;

 proc transpose data=scores out=scores1(rename=(_name_=Variable_label_=VarLabel));run;
 proc sort data=scores1;
    by variable;
 run;

 proc sort data=vclus_out3;
    by variable;
 run;

 data score_coef;
    merge scores1 vclus_out3;
    by variable
 run;

 proc sort data=score_coef;
    by cluster;
 run;

 proc sort data=clusters&ncl1;
     by variable;
 run;

 data score_coef1;
   length score_coef 4.;
  set score_coef;
  score_coef=sum(of clus1-clus&ncl2);
 run;

 proc sort data=score_coef1;
   by variable;
```

```
 run;

 data final_out;
   length variable $30. ;
   merge vclus_out3 clusters&ncl1(drop=cluster controlvar) score_coef1;
   by variable;
   if cluster^=0;
 run;


proc sort data=final_out;
  by cluster;
run;

proc rank data=final_out ties=low out=final_out;
  by cluster;
  var rsquareratio;
  ranks r2_rank;
run;
```

The following is a sample of the output from variable clustering:

variable	Cluster	score_coef	r2_rank	OWN_CLUS
AD_MAG_D90_ORD_AMT	1	0.042	20	0.863056839
AD_MAG_D90_PAY_AMT_CASH	1	0.04074	23	0.812197296
AD_MAG_D90_PAY_AMT_TOTAL	1	0.04164	22	0.848346051
AD_MAG_D90_PAY_COUNT_CASH	1	0.04178	21	0.8541178
AD_MAG_D90_PAY_COUNT_TOTAL	1	0.04224	19	0.873119125
AD_MAG_TOT_BILL_LVL_INVOICE	1	0.04378	16	0.937981186

STEPS 5: CREATE SUBSET VARIABLE TABLE

The final variable table documents the entire process. Variables that survived all 4 screens are preserved on the final data set. The table shows statistics from ANOVA, bagging, and clustering.

The following is a sample of the variable table:

variable	DF	MS	FValue	ProbF	Bag #	Bag %	Cluster	score_coef	r2_ rank	OWN_ CLUS	NEXT_ CLUS	R2 _RATIO
AD_MAG_...	1	3101.44	11.4	0.0007	1	0.1	1	0.045	1	0.977	0.358	0.035
AD_MER_...	1	180967.00	19.7	<.0001	2	0.2	3	0.285	1	0.848	0.349	0.233

The screened data set is now a workable size for the statistician. The following are actual statistics from a recent model development effort at Alliant Cooperative Data Solutions:

Unscreened Data Set:	Screened Data Set:
N=317,581	N=317,581
# Variables: 6,513	# Variables: 369
Physical Size: 4,879 MB	Physical Size: 416.6 MB

SOME ISSUES FOR USING PROC LOGISTIC TO FIT A GENERALIZED LOGISTIC MODEL

In direct mail applications, it's typical to code the non-responders as 0 and code the responders 1. Most modelers then use the DES option to model the responders. However, using the DES option for multiple responders can be confusing. For the example in the case study I used REF=FIRST. This maintains the codification for 0 as the reference group without switching the order between the response categories and the logits.

If you forget to use the option LINK=GLOGIT you will end up with a proportional hazards model. The proportional hazards model fits 1 slope for every predictor variable with multiple intercepts. This type of model is useful when the predictors have the same effect across the different levels of responses. However, in direct response applications the effect of a variable modeling a customer's propensity to return a product is quite different from the effect of a variable modeling a customer's propensity to bad debt on a product.

Notice that in the example, I used the SCORE statement to score the development data set and a separate validation data set. The SCORE data set includes all of the fields on the input data set. By default, it adds variables corresponding to the estimated probabilities. In the example above, the scored data set would include all of the input variables as well as P_0, P_10, P_20, P_30, and P_40.

For simple logistic regression, modelers often use the OUTPUT statement. However, the OUTPUT statement produces 1 record for each degree of freedom for the response variable. In the example above, if the original data set had 300,000 records. The data set produced by the OUTPUT statement would include 1,200,000 records.

In production environments, the estimated coefficient file produced from the OUTEST option is often used to score future sets of data. For generalized logistic regression, a '_' followed by the level of the dependent variable is appended to the name of the predictor variable. Be careful that the length of the name of predictor variable plus the '_' followed by the level of the dependent variable is less than 32 bytes. SAS preserves the '_' and the level of the dependent variable but it will truncate the name of the predictor variable to ensure that name of the field is less than 32 bytes. This can be problematic for production scoring routines that match the variable name to the estimated coefficient.

We have found it useful to convert the coefficient file to a text file that can be used in a data step. Upon request, I will send you a macro that performs that conversion.

CONCLUSION:

Generalized logistic regression has proven to be a useful tool for estimating probabilities to be used in scoring models for direct response clients. Stepwise selection of variables for the generalized logistic regression model is too slow to be useful on very large data sets. Variable selection on very large data sets needs to be performed via custom routines. The SAS system facilitates the building of such routines.

The solution that I presented today should give you ideas on how to customize your own selection routine.

REFERENCES:

McFadden, D. (1974), "Conditional Logit Analysis of Qualitative Choice Behaviour" in Frontiers in Econometrics, edited by P. Zarembka, New York: Academic Press.

Hosmer DW, Lemeshow S. Applied Logistic Regression. New York: John Wiley & Sons; 1989.

Breiman, L. (1996), "Bagging Predictors" Machine Learning, 26:2, 123-140

Freidman, J. and P. Hall (1999) "On Bagging and Nonlinear Estimation" *www.stat.stanford.edu/~jhf*

Leo Breiman's homepage www.stat.berkeley.edu/users/breiman

ACKNOWLEDGMENTS:

I'd like to extend my thanks to Russell Greenberg of Alliant Cooperative Data Solutions for developing the bagging routine for this process

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

John A. Cherrie
Alliant Cooperative Data Solutions
301 Fields Lane
Brewster, NY 10509
Work Phone: (435) 647-5752
E-mail: jcherrie@alliantdata.com
Web: http://www.alliantdata.com/contact_directory.asp