Paper 150-2011

SAS® Base Implementation of Information Theoretic Feature Selection for Neural Networks

Martin Jetton, Kronos, Inc, Beaverton, OR, USA

ABSTRACT

In "Neural Network Modeling using SAS® Enterprise Miner", Matignon lists' one of the disadvantages of neural network modeling as 'No Universal Input Variable Selection Routine" (page 152). Dr David Scarborough and Bjorn Chambless (2001) established the use of Information Theoretic Feature selection in pre-employment application neural network modeling. Information theoretic entropy provides a convenient metric for estimating the difference between distributions. This paper describes the development of a feature selection algorithm implemented in BASE SAS(R).

INTRODUCTION

In neural network terminology, inputs are predictor variables and outputs are criterion variables. The process of identifying the subset of predictor variables to be used in a model is called feature selection. When the set of predictor variables is large, with many of the individual variables having predictive value, it is best to reduce the set of predictors as much as possible. There are several reasons to reduce the number of inputs to a parsimonious group. First, not all potential inputs have significant predictive value. The use of variables with little or no predictive value as inputs can be seen as adding noise and requires the neural network the additional burden of learning to ignore these inputs and may degrade model performance. This requires additional training time and perhaps additional neural resources. Second, assuming a fully connected feed-forward neural network model, the complexity (as measured in the number of network connections) increases geometrically with the number of inputs. As complexity increases so does training time along with the network's susceptibility to over-training. It is therefore desirable to eliminate inputs with less predictive power in favor of a less complex neural network model.

Feature selection techniques may be viewed as falling into one of two categories filters and wrappers. Wrappers use the relationship between model performance and IVs directly by iteratively experimenting with different subsets of IVs. One weakness of this approach is the potentially large number of IVs subsets and the expense of creating and testing a complete model for each of these subsets in addition to the problem of non-determinism within the modeling process.

Filters are divorced from the specific modeling technique. These methods analyze the relationship between sets of IVs and DVs using methods independent of those used to develop the model.

INFORMATION THEORETIC FEATURE SELECTION

For our prediction model, information theoretic methods were employed to determine the subset of input variables which maximized information transmission between the IVs and the DVs. This methodology relies on the statistical theory of independent events, where events $p_1, p_2, ..., p_n$ are considered *statistically independent* if and only if the probability P, that they all occur on a given trial is

$$P = \prod_{i=1}^{n} p_i$$

Conversely, the measurement of how much a joint distribution of probabilities differs from the independence distribution can be used as a measure of the *statistical dependence* of the random events.

Information theoretic *entropy* provides a convenient metric for estimating the difference between distributions. The entropy, $H(X)$ (measured in bits) of the distribution of a discrete random variable X with n states is

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the probability of state i.

Its utility springs from the fact that entropy is maximized when a distribution is uniform. For example, figure 1 shows a graph of the entropies of single variable, discrete 2-state distributions and the probabilities vary.

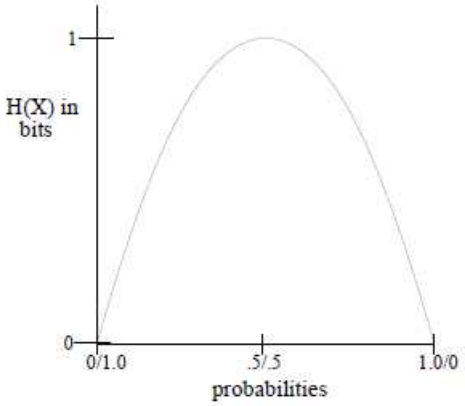

Figure 1 -- Entropy for X, a 2 state discrete distribution

Similarly, for a multivariate distribution constrained by specified marginal distributions, the distribution which maximizes entropy will be the independence distribution. Therefore, given a joint distribution with fixed marginals, the distribution which minimizes entropy will be the distribution for which the variables are completely dependent.

Dependence can be viewed as constraint between variables and as constraint is reduced, entropy increases. Information theoretic analysis of a distribution is then the measurement of constraint.(see figure 2).

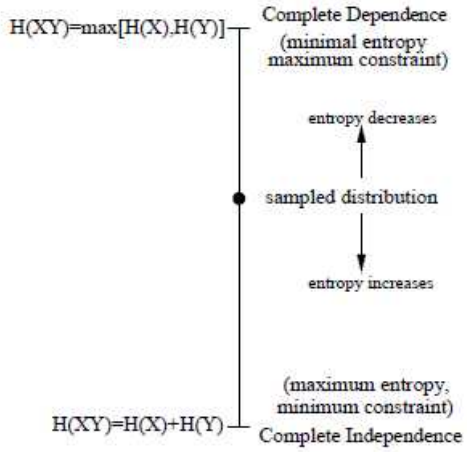

Figure 2 -- Constraint vs. entropy

Assuming some constraint between variables, a sampled distribution will lie somewhere between complete dependence and independence and will have a measurable entropy. If we are analyzing the joint distribution of the variables X and Y, the entropy for this sampled distribution is $H(XY)$. The entropies of the variables X and Y measured separately are $H(X)$ and $H(Y)$ and are computed using the marginals of the joint distribution.

Since $H(X)$ and $H(Y)$ are calculated from the marginals and entropy is logarithmic $H(X)+H(Y)=H(XY)$ if there is no constraint between X and Y. Or

$$H(XY)=H(X)+H(Y) \text{ } \textit{if and only if X and Y are independent}$$

This equality would indicate that there is absolutely no relationship between X and Y and the joint distribution of the variables is the independence distribution.

Information transmission T is the measure of the distance between distributions along the continuum shown in figure 2. For discrete random variables X and Y, $T(X:Y)$ the information transmission between X and Y, is computed

$$T(X:Y) = H(X) + H(Y) - H(XY)$$

$T(X:Y)$ is the difference between the entropies of the independence distribution and the sampled joint distribution. The degree of dependence between X and Y can therefore be computed by measuring information transmission. A small value for $T(X:Y)$ indicates the variables X and Y are nearly independent, whereas a large value suggests a high degree of interaction.

In a directed system, such as a predictive model, the measure of information transmission between the distribution of an independent variable X and a dependent variable Y can be used to gauge the predictive value of X.

Ideally, the goal is to find a subset S of the independent variables V such that, for the set of dependent variables D.

$$T(D:V) \approx T(D:S)$$

However, as discussed, the modeling technique to be employed may limit the cardinality of S so the filtering process should be guided by the following considerations:

1. if S' is any subset of V smaller than S, then $T(D:S')$ is significantly smaller than $T(D:S)$.
2. if S' is any subset of V larger than S, then $T(D:S')$ is not significantly larger than $T(D:S)$

Since information theoretic transmission only measures the degree of difference between distributions of variables, not the nature of the difference, the technique can be considered 'model free'. This property allows the methodology to work as an effective filter regardless of the subsequent modeling techniques employed.

Such filtering methods fall into the general category of *dependency analysis* (Conant) which in turn have grown out of the discrete modeling work of Ashby and Klir.

ONE, TWO AND THREE WAY TRANSMISSION

With the ability of neural networks to capture the complexity of inputs to outputs, the implementation of information theoretic feature selection considers evaluating each input on their own (one-way transmission to criteria), with another input (two-way transmission to criteria) and with two other inputs (three-way transmission to criteria). Higher dimensionality (4, 5, etc way transmissions) only make sense when the criteria dataset consists of 100,000 observations (or even more). Four-way and above matrices start to become very sparse in smaller datasets. To understand the implementation let's look at some examples of each n-way transmission calculations.

ONE-WAY EXAMPLE

One-way implementation of information theoretic feature selection looks a lot like one-way contingency table tests for independence. It is a test for independence but with a different formula.

Figure 3 -- Looking at inputs and criteria as a 'communication' channel. What does 'input' tell us about the 'criteria?'

An illustrative example of independence to help understand the calculation of entropy and transmission from observed data. H(Input) is the marginal distribution entropy of its marginal percentages as $p_A*\log_2 p_A + p_B*\log_2 p_B$. In this case just two categorical options: A and B, for input. H(Criteria) is similar. While H(Input*Criteria) is the entropy

of the observed percentage distribution. Using the formula above for the transmission between Input and Criteria in this channel: T(Input:Criteria) = H(Input) + H(Criteria) – H(Input*Criteria). We can see in this example where Input and Criteria are independent that the transmission is zero (0). This is not what we are looking for in information theoretic feature selection.

			Criteria		Criteria	
Input	2,500	25%	625	1,875	Input	6% 19%
	7,500	75%	1,875	5,625		19% 56%
			25%	75%		
			2,500	7,500	10,000	

T(Input:Criteria) = H(Input) + H(Criteria) - H(Input*Criteria)

H(Input) = H(.25)+H(.75) =
0.81
H(Criteria) = H(.25)+H(.75) =
0.81
H(Input*Criteria) = H(.06)+H(.19) +H(.19)+H(.56)=
1.6226

T(Input:Criteria) = 0 --> *independence*

Figure 4 -- Example of independence

To illustrate the more useful relationship we are looking for in feature selection, here are the same 10,000 observations but when input and criteria are not independent. You can see that the transmission T(Input:Criteria) is 0.0871. Using the relationship $L^2 = 1.3863 * n * T()$ (Krippendorff, page 53), we can test the significance against a χ^2 distribution with, in this case, 1 degree of freedom. In this case with 10,000 observations T() + 0.0871 is statistically significantly different from zero (aka not independent).

Also shown in figure 5 is the alternative calculation implemented in SAS (Krippendorf, page 24)

			Criteria		H(Input*Criteria) Criteria		H(Input:Criteria) Criteria	
Input	2,000	20%	1,000	1,000	Input	10% 10%	Input	4% 16%
	8,000	80%	1,000	7,000		10% 70%		16% 64%
			20%	80%		*observed*		*expected*
			2,000	8,000	10,000			

T(Input:Criteria) = H(Input) + H(Criteria) - H(Input*Criteria)

H(Input) = H(.2)+H(.8) =
0.72
H(Criteria) = H(.2)+H(.8) =
0.72
H(Input*Criteria) = H(.1)+H(.1) +H(.1)+H(.7)=
1.3568

T(Input:Criteria) = 0.0871 --> *not independent it's greater than 0*

H(Input:Criteria) = H(.04)+H(.16) +H(.16)+H(.64) =
1.4439

*T(Input:Criteria) = H(Input:Criteria) - H(Input*Criteria)*
H(observed) - H(expected)
T(Input:Criteria) = 0.0871

Figure 5 -- Example of non-independence and an alternative calculation.

Now given that we have determined independence for the individual inputs to criteria relationship, the next step is to evaluate two way or multi input relationships to criteria. Given the complexity that neural networks can capture, these multi way significant relationships hopefully will critical to the selection of inputs.

The SAS® code is implemented as a macro to iterate throw a set of input variables one at a time:

```
%macro one_way (iv);

proc sql;
create table temp1 as
select &dv,
       &iv,
       count(*) as cnt_act
       from &dataset
       group by &dv,&iv;

create table temp2 as select &iv,  sum(cnt_act) as iv_count  from temp1 group by &iv;

quit;

data temp3; set temp2;do i = &&list&dv; &dv = i; output temp3; end;  run;

proc sort data=frq_&dv; by &dv; run;
proc sort data=temp3; by &dv; run;

data temp3 ;
 merge temp3 frq_&dv;
  by &dv;
  iv_percent = iv_count / &cnt_ttl ;
  dv_percent  = percent/100;
  drop percent;
  dv_count = count;
  drop count;
run;

proc sort data=temp1; by &dv &iv; run;
proc sort data=temp3; by &dv &iv; run;

data temp4 (keep = dv_var iv_var log_p);
merge temp3 end=end_file temp1;
 by &dv &iv ;
 retain log_p 0;
 q = iv_percent * dv_percent;
 p = cnt_act / &cnt_ttl;
 if p > 0 then log_p = p * log2(p/q) + log_p;
 length dv_var $30;
 length iv_var $30;
 dv_var = resolve("&dv");
 iv_var = resolve("&iv");
 if end_file then output;
  run;

  data one_ways;
  set one_ways temp4;
  samplesize= &cnt_ttl ;
  dv_var_logp = &&&dv.logp;
  run;
%mend;

proc sql;
select count(*) into : cnt_ttl
       from &dataset;
quit;
```

```
data one_ways;run;

data _null_;set oneways;
 call execute('%one_way(iv=' || iv1 ||')' );
run;

proc sort data=one_ways; by descending log_p ;run;
PROC EXPORT DATA= WORK.one_ways OUTFILE= "Desktop\OneWayTransmissions.csv"
DBMS=CSV REPLACE; RUN;
```

TWO-WAY EXAMPLE

The two-way implementation of information theoretic feature selection begins to show the distinction from contingency table tests for independence.

Figure 6 -- Looking at inputs and criteria as a 'communication' channel. What do the two 'inputs' tell us about the 'criteria?'

Using the first example when input 1 was shown to be independent of criteria we can illustrate the value of a second input relative to the criteria. When the second input (input 2) is used to split the criteria across input 2's options (A and B), we see an interesting structure to the contingency table. There are no observations in the following combinations: input 1 = "A" & input 2 = "A" & criteria ="B", input 1 = "B" & input 2 = "A" & criteria ="A", input 1 = "A" & input 2 = "B" & criteria ="A", and input 1 = "B" & input 2 = "B" & criteria ="A". By adding in this second input the combination of input 1 and input 2 starts to get interesting.

				Criteria A	Criteria B
Input 1	A	2,500	25%	625	1,875
	B	7,500	75%	1,875	5,625
				25%	75%
				2,500	7,500	10,000

				Input 2 = A: Criteria A	Input 2 = A: Criteria B	Input 2 = B: Criteria A	Input 2 = B: Criteria B
Input 1	A	2,500	25%	625	-	-	1,875
	B	7,500	75%	-	5,625	1,875	-
				6%	*56%*	*19%*	*19%*
				625	5,625	1,875	1,875	10,000

Figure 7 -- When adding a second input to the model, we see some interesting structure in the data.

Taking this interesting structure and calculating the observed percentages and the expected percentages (based upon the marginal for input 1 and input 2 together), we can apply information theoretic measurements to the information in this combined, input 1 / input 2 situation.

Observed:

		Input 2			
		A		B	
		Criteria		Criteria	
		A	B	A	B
Input 1	A	6%	0%	0%	19%
	B	0%	56%	19%	0%

Expected

		Input 2			
		A		B	
		Criteria		Criteria	
		A	B	A	B
Input 1	A	2%	14%	5%	5%
	B	5%	42%	14%	14%

Figure 8 -- Observed and Expected distributions for input 1 and input 2 to criteria.

We calculate the H(observed) from the distribution of observed and the H(expected) from the distribution of expected probability and see that the difference, transmission (T()) is 0.8113. This is a huge transmission approaching 1.0 (the max for T()).

H(observed) = 1.6226

H(expected) = 2.4338

T(Input1*Input 2:Criteria) = 0.8113

Figure 9 -- H(observed) and H(expected) used to calculate the transmission for this situation.

		Input 2	
		A	B
Input 1	A	625	1,875
	B	1,875	5,625

Figure 10 -- for those of you who are interested, collapsing criteria out of the contingency table shows that input 1 and input 2 are independent (same table analyzed above)

The SAS® code is implemented as a macro to iterate throw a set of input variable pairs one pair at a time:

```
%macro two_way (iv1,iv2);

proc sql;
create table temp1 as
select &dv,&iv1,&iv2,count(*) as cnt_act
       from &dataset
       group by &dv,
              &iv1,&iv2;

create table temp2 as
select &iv1, &iv2, sum(cnt_act) as iv_count
       from temp1
       group by &iv1,&iv2;
quit;

data temp3; set temp2; do i = &&list&dv;  &dv = i; output temp3;  end; run;

proc sort data=frq_&dv; by &dv; run;
proc sort data=temp3; by &dv; run;

data temp3 ;
 merge temp3 frq_&dv;
  by &dv;
  iv_percent = iv_count / &cnt_ttl ;
  dv_percent  = percent/100;
  drop percent;
```

```
  dv_count = count;
  drop count;
run;

proc sort data=temp1; by &dv &iv1  &iv2; run;
proc sort data=temp3; by &dv &iv1  &iv2; run;

data temp4 (keep = dv_var iv1_var  iv2_var log_p);
merge temp3 end=end_file temp1;
 by &dv &iv1 &iv2 ;
 retain log_p 0;
 q = iv_percent * dv_percent;
 p = cnt_act / &cnt_ttl;
 if p > 0 then log_p = p * log2(p/q) + log_p;
 length dv_var $30;
 length iv1_var $30;length iv2_var $30;
 dv_var = resolve("&dv");
 iv1_var = resolve("&iv1");
 iv2_var = resolve("&iv2");
 if end_file then output;
  run;

  data two_ways;
  set two_ways temp4;
  run;

%mend;

data two_ways;  run;

proc sql;
select count(*) into : cnt_ttl   from &dataset;
quit;


data _null_;
set twoways;
call execute('%two_way(iv1=' || iv1 || ' , iv2=' || iv2 || ')' );
run;

proc sort data=two_ways; by descending log_p ;run;

PROC EXPORT DATA= WORK.two_ways
            OUTFILE= "Desktop\TwoWayTransmissions.csv"
            DBMS=CSV REPLACE;
RUN;
```

THREE-WAY TRANSMISSIONS

The SAS® code is implemented as a macro to iterate throw a set of input variables one set of triplets at a time in the macro:

```
%macro thre_way (iv1,iv2,iv3,n);
proc sql;
create table temp1 as
select &dv,  &iv1,  &iv2,  &iv3,  count(*) as cnt_act
       from &dataset group by &dv,&iv1,&iv2,&iv3;

create table temp2 as
select &iv1,&iv2,   &iv3,  sum(cnt_act) as iv_count
       from temp1 group by &iv1,&iv2,&iv3;
quit;
```

```
data temp3; set temp2; do i = &&list&dv; &dv = i; output temp3; end; run;

proc sort data=frq_&dv; by &dv; run;
proc sort data=temp3; by &dv; run;

data temp3 ;
 merge temp3 frq_&dv;
  by &dv;
  iv_percent = iv_count / &cnt_ttl ;
  dv_percent  = percent/100;
  drop percent;
  dv_count = count;
  drop count;
run;

proc sort data=temp1; by &dv &iv1  &iv2 &iv3; run;
proc sort data=temp3; by &dv &iv1  &iv2 &iv3; run;

data temp4 (keep = dv_var iv1_var  iv2_var iv3_var log_p);
merge temp3 end=end_file temp1;
 by &dv &iv1 &iv2 &iv3;
 retain log_p 0;
 q = iv_percent * dv_percent;
 p = cnt_act / &cnt_ttl;
 if p > 0 then log_p = p * log2(p/q) + log_p;
 length dv_var $30;
 length iv1_var $30;length iv2_var $30  iv3_var $30;
 dv_var = resolve("&dv");
 iv1_var = resolve("&iv1");
 iv2_var = resolve("&iv2");
 iv3_var = resolve("&iv3");
 if end_file then output;
  run;

proc append  base=three_ways data= temp4 force; run;

%mend;

 data three_ways;
 length dv_var $30;
 length iv1_var $30;length iv2_var $30  iv3_var $30;
 dv_var = "&dv";
 iv1_var = "dummy";
 iv2_var = "dummy";
 iv3_var = "dummy";
 log_p = -0.99999;
run;

proc sql;select count(*) into : cnt_ttl from &dataset;quit;


data _null_;
set threeways;
call execute('%thre_way(iv1=' || iv1 || ' , iv2=' || iv2 || ' , iv3=' || iv3 || ' ,
n=' || _n_ ||')' );
run;

proc sort data=three_ways; by descending log_p ;run;

PROC EXPORT DATA= WORK.Three_ways
            OUTFILE= "Desktop\ThreeWayTransmissions.csv"
            DBMS=CSV REPLACE;
RUN;
```

SAS® IMPLEMENTATION

BASE SAS® implementation consists of 6 scripts:

1) Processing the file of data for columns classification into categorical / continuous and IV / DV (independent / dependent) variables. This creates 3 tables for automated processing of IV to DV relationships.
2) Adding/creating categorical variables from continuous variables on the data table of interest.
3) Processing one way transmission (one IV to DV) relationship & printing frequency tables of top 10 (PROC FREQs not shown in code below).
4) Processing two way transmission (two IV to DV) relationships & printing frequency tables of top 20 (PROC FREQs not shown in code below).
5) Processing three way transmission (three IV to DV) relationships & creating/printing frequency tables of top 40 (PROC FREQs not shown in code below).
6) Using mosaic plots to present to clients. (not included in this paper but SAS® code is available upon request)

Step 1 consists of the following BASE SAS® code that reads the columns from the dataset of interest, the user needs to put the data in MS Excel to strip out the column names and add indicators of IV / DV's of Y/N and categorical / continuous as Y/N. This dataset is then pasted back into the SAS® code to be used in creating the 'cols' dataset shown.

```
/* run this with the 'data=' dataset set to the one to be analyzed*/
libname workdesk "Desktop";
%let dataset = workdesk.inputs_criteria;

proc datasets library=work details; contents data=&dataset out=&dataset._cols varnum;
run; quit;

proc print data=&dataset._cols;
var name length type;
run;

/* move the dataset over to Xcel to create the IVs and DVS */

data cols;
informat vars $30. ivdv $1. contin $1.;
input vars ivdv contin;
if ivdv="Y";
varnum = _n_;
matchnum = _n_;
cards;
Quest1       Y      N
…
Quest86      Y      N
Factor1      Y      Y
…
Factor8      Y      Y
Scoring1     Y      Y
…
Scoring6     Y      Y
IRT1   Y     Y
…
IRT7   Y     Y
Criteria     N      N
;
run;

proc freq data=cols; table ivdv * contin; run;

%let numcols = 107;
```

```
data oneways;
Set cols (keep = vars contin varnum matchnum rename = (varnum =  iv1_varnum  vars =
iv1 contin  = iv1_contin ));
if iv1_contin ="Y" then iv1 = compress(iv1 ||'_con');
run;

data cols2 (drop=i);
set cols;
do i=1 to &numcols;
 matchnum = i;
 *if contin ="Y" then vars = compress(vars ||'_con');
 output cols2;
 end;
 run;

proc sort data=cols; by matchnum; run;
proc sort data=cols2; by matchnum; run;

data twoways;
merge cols2 (keep = vars contin varnum matchnum rename = ( varnum =  iv1_varnum contin
= iv1_contin vars = iv1 ))
             cols (keep = vars contin varnum matchnum rename = (varnum =  iv2_varnum
vars = iv2 contin  = iv2_contin ));
by matchnum;
if iv1_contin ="Y" then iv1 = compress(iv1 ||'_con');
if first.matchnum and iv2_contin ="Y" then iv2 = compress(iv2 ||'_con');
if iv1 = iv2 then delete;
if iv2_varnum > iv1_varnum;
run;

data twoways2 (drop=i);
set twoways;
do i=1 to &numcols;
 matchnum = i;
 output twoways2;
 end;
 run;

proc sort data=twoways2; by matchnum; run;

data threeways;
merge twoways2 (keep = matchnum iv2_varnum  iv2 iv2_contin iv1_varnum iv1_contin iv1)
             cols (keep = vars contin varnum matchnum rename = ( varnum =  iv3_varnum
contin  = iv3_contin vars = iv3 ));
by matchnum;
if first.matchnum and iv3_contin ="Y" then iv3 = compress(iv3 ||'_con');
if iv3 = iv2 or iv3 = iv1 then delete;
if iv3_varnum > iv2_varnum;
run;
```

Step 2 consists of 2 macros; one for the DV categorical variable to create summary frequency tables to be use in the processing and one to create and add the IV categorical versions of the continuous variables on the dataset of interest. The IV's that are continuous need to be categorical for analysis purposes. To do this PROC UNIVARIATE is used to identify the continuous variables 20th, 40th, 60th and 80th percentiles. The dataset created in step 1, cols, and the indicators of categorical/continuous ('contin') is used to execute the macro.

```
/* CATEGORICAL VALUES MACROS */

options nomlogic nomrecall nosymbolgen nomprint;
options nosource nosource2 nonotes;
```

```
%MACRO CAT_varS (var_NAME);
%global list&var_name;
%global &var_name.logp;

proc freq data=&dataset noprint; table &var_name /out=frq_&var_name; run;

proc sort data=frq_&var_name; by &var_name; run;

data _null_;
set frq_&var_name end=end_file;
by &var_name ;
retain var_list _char_ "                              ";
if _n_ = 1 then spliter=" "; else spliter="," ;
var_list = trim(var_list) || spliter || put(&var_name,3.);
if end_file then call symput(resolve('list&var_name'),var_list);
retain log_p 0;
log_p = (percent/100) * log2(percent/100) + log_p;
if end_file then call symput(resolve('&var_name.logp'),log_p);
run;

data &dataset._cols; set &dataset._cols;
if name = "&var_name" then p_log_p = &&&var_name.logp; run;
%mend;
```

```
%MACRO CONT_varS (var_NAME);

proc univariate data=&dataset noprint; var &var_name;
output out=out_p
       pctlpre=P_ pctlpts=20 to 80 by 20;
run;

data _null_;
set out_p;
call symput('p20_cut',p_20);
call symput('p40_cut',p_40);
call symput('p60_cut',p_60);
call symput('p80_cut',p_80);
run;

data &dataset;
set &dataset;
if &var_name < -99999 then delete;
else if &var_name < &p20_cut then &var_name._con = 1;
else if &var_name < &p40_cut then &var_name._con = 2;
else if &var_name < &p60_cut then &var_name._con = 3;
else if &var_name < &p80_cut then &var_name._con = 4;
else &var_name._con = 5;
run;

%mend;
```

```
%let dv = criteria;

%cat_vars(&dv); %put &&&dv.logp;

data _null_;
set cols;
if contin = "Y" then call execute('%cont_vars(var_NAME=' || vars ||')' );
run;
```

CONCLUSION

This paper described the implementation of information theoretic entropy feature selection algorithm implemented in BASE SAS®. Information provides a convenient metric for estimating the difference between distributions. When the set of predictor variables is large, with many of the individual variables having predictive value, it is best to reduce the set of predictors as much as possible. The inputs with the strongest relationship in singletons, dyads and triads inputs to a neural network will enable the identification of a parsimonious set of inputs for use in predictive modeling.

REFERENCES

Ashby, F. Gregory, 1992, *Multidimensional Models of Perception and Cognition*, Psychology Press

Chambless, B. and Scarborough, D., 2001, "Information theoretic feature selection for a neural behavioral model". *Proceedings of the International Joint Conference on Neural Networks of the IEEE*, Washington D.C. US Patent #20100287111

Conant, R., "Extended Dependency Analysis of Large Systems", *International Journal General Systems*, vol 14, pp 97-141, 1988

Cover, Thomas and Thomas, Jay, 2006, *Elements of Information Theory, 2nd Edition*, Hoboken, NJ, Wiley

Krippendorf, Klaus, 1986, *Information Theory, Structural Models for Qualitative Data*, Newbury Park, CA, Sage University Paper

Klir, George, 2006, *Uncertainty and Information, Foundations of Generalized Information Theory*, Hoboken, NJ, Wiley

Klir, George and Elias, Doug, 2003, *Architecture of systems problem solving 2nd edition*, Kluwer Academic/Plenum Publisher,

Matignon, Randall, 2005, *Neural Network Modeling using SAS Enterprise Miner*, Page 152, Authorhouse

Shannon, C.E. "A mathematical theory of communication", *Bell Systems Technical Journal*, vol. 27, 1948

RECOMMENDED READING

- Matignon, Randall, 2005, *Neural Network Modeling using SAS Enterprise Miner*, Authorhouse

CONTACT INFORMATION

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

Name:	Martin Jetton
Enterprise:	Kronos, Inc
Address:	9525 SW Gemini Dr
City, State ZIP:	Beaverton, OR 97008
Work Phone:	(503) 596-3181
Fax:	(503) 596-3200
E-mail:	Martin.Jetton@Kronos.com