How to detect potential customers

Achim Lewandowski
FORWISS- Bayerisches Forschungszentrum für wissensbasierte Systeme
Am Weichselgarten 7, D-91058 Erlangen
Phone: +49-9131-691257, Fax: +49-9131-691185
http://www.forwiss.uni-erlangen.de/~alewand
email:alewand@forwiss.de

ABSTRACT: An important task for a direct mailing company is to detect potential customers to avoid unnecessary and unwanted mailing. The aim of this document is to illustrate the way the author has chosen to cope with this problem in the case of the COIL CHALLENGE 2000 data set.

KEYWORDS: direct mailing, frequency analysis, variable selection, scoring systems, neural network

BASIC IDEAS

The COIL data set encloses 85 possible, sometimes highly correlated input variables. To find a reliable subset of variables is the first (maybe *the only*) important task. I decided (after trying trees and nearest neighbour methods) to build up a scoring system whereby only important variables should be integrated. Originally my intention was to feed these variables as inputs to a neural network. But after the announcement of the true policy owners it has become clear that the application of neural networks isn't necessary, the scoring system itself could be used instead and, even more, would have performed better.

To keep things easy I restricted the score of a potential customer to be the sum of partial scores, whereby each partial score should belong just to the value of a single variable. If a selected variable has a specific value, the partial score should be mainly determined by the ratio of good (G) to bad (B) customers for this combination. I rejected the first idea to take the quotient $q = G/(G+B)$, because it doesn't depend on the number of customers. For example, the combination (G=1, B=2) would be judged the same as (G=100, B=200). As the goal was to find the upper 20 percent of potential customers, I used the proportion in the whole training set as a base line (q=348/5822).

To compute the partial score of a variable V with value w, one has to count the good and bad customers with $V = w$ and compute the test statistic for the hypothesis „The probability in this subset to be a good customer is q". The resulting test statistic is

$$T = \frac{G - q(G+B))}{\sqrt{q(1-q)(G+B)}}$$

and is used as a partial score, if a customer has value w in variable V. For high values of the test statistic it is more probable to find good customers in this subset, in comparison to the whole sample.

FINDING THE BEST VARIABLES

It was mentioned earlier that the total score of a customer should be the sum of its individual partial scores. The next task is to find convenient variables which should be included in the scoring system. For this process I used a stepwise procedure, which adds one variable per step. The procedure consists of the following steps:

1. Initialize the set of the already chosen variables A with $A = \varnothing$

2. Repeat the steps 2a)-2d) 100 times

   a) Generate a training set (5322 examples) and a test set (500 examples)

   b) Compute the partial scores for every variable (using only the training set)

   c) Compute the sum of partial scores of the variables in A for the customers in the test set

   d) Repeat d1) - d3)for every variable in a candidate set (can be always V1-V85))

      d1) Add the partial scores of the chosen variable to the scores computed in 2c)

      d2) Find the upper 20 percent with the highest scores

      d3) Compute the percentage of recognized good customers (in relation to all good customers in the test set)

3. Compute the mean of recognized good customers for all candidate variables

4. If there was a significant improvement: Include the best variable in A and start over with 2.

For my submission I used a simpler variation, each variable could only be included once. The possibility of adding the partial scores of a single variable several times allows to model a more general class of scoring systems. Additionally I'm not sure if it's wise to use different partitions for each new run of step 2. Maybe there is some kind of overfitting if always the same 100 partitions are used. For my submission I generated always new training and test sets. The performance from variable to variable was measured on different test sets.

THE RESULTS

I started two runs. The first run produced A={59, 47, 25, 82, 65}. The scoring system with these variables would have recognized 116 policy owners. Another run resulted in A={59, 47, 1, 68, 61, 76, 82} with 115 policy owners in the chosen set of 800 customers. Most information is included in the variables {59, 47}. Using these variables alone would have guaranteed to recognize 113 policy owners. With A={59, 47, 14, 18, 32} it is even possible to get 129 correct cases. Unfortunately I have discovered this fact *after* the announcement of the true policy owners.

I generated a new run (which selected different variables as in my submission) to show the development of the mean percentage and the number of the recognized policy owners in the evaluation set:

Added Var.	59	47	1	16	82	47	78	78	78	79
Mean Percentage	41.5	53.8	55.2	54.6	54.6	55.8	56.2	56.6	56.7	56.7
Number of True Policy owners	94	113	120	120	115	116	118	118	118	118

In every run V59 was the first and V47 the second variable, which was chosen. Also V82 was included sooner or later.

I decided to take the second run with $A = \{59, 47, 1, 68, 61, 76, 82\}$ and used the chosen variables as inputs for a neural network. I recoded two variables, V1 and V47. In the case of V1 I collected most values in a single value and for V47 I changed the order of the values. As I didn't get a single network to produce the same results as my scoring system, I tried Leo Breiman's arcing algorithm. This algorithm generates a sequence of samples whereby cases which haven't been correctly classified, get higher probability to be included in the next sample. For each sample a classifier is built and the results of all classifiers are combined.

During my experiments I had the impression, that the performance of the arcing algorithm was slightly better on cross-validation sets. Finally I decided to submit the results of two runs. The first run would have recognized 115 customers, the same as the scoring system, and the second run 111 customers. I computed the mean of the forecasted probabilities. The influence of the 111-customers-run was stronger, I finally got a score of 111 for my submission. Five other runs, which I performed later, had values of 113, 110, 113, 113, 113. Therefore the expected value, when using the arcing algorithm, should be around 113. This is the same value which would have been possible using my scoring system with V47 and V59, although not the with the same chosen customers.

DESCRIPTION OF POLICY OWNERS

It's very difficult (maybe impossible) to give an interpretation of the result of an ensemble of neural networks. It is clear that the most important variables are V47 and V59. If one chooses all 777 cases with V47=6 and (V59=3 or V59=4 or V59=5) one has caught 112 good customers. If the remaining 23 cases are drawn randomly, one gets about 113 recognized policy owners. I used additionally the variables {1, 68, 61, 76, 82} and got 111 policy owners in my selected set (the score rises to 115 when using the scoring system). It seems plausible not to speak of assured characteristics of policy owners, if one doesn't outperform the value of 113 clearly. So my simple description of the top-20-percent-customers is

> The top-customer has a car policy contribution of 6 and a fire policy contribution of 3 or 4 or 5.

Besides the fact that my ensemble of neural nets has chosen all cases with V47=6 and V59=4, it's too difficult to explain how the rest is composed. Of course it's possible to print a list of scores for all customers but with 7 used variables a lot of combinations exist and I don't believe that a clear pattern will be revealed. The scoring system could describe the contribution of each variable to the score, but without the explicit computation of the global score, it will be difficult to decide which of two customer types should be preferred.