View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Erasmus University Digital Repository

Modeling Brand Choice using Boosted and Stacked Neural Networks

Rob Potharst, Michiel van Rijthoven and Michiel van Wezel
Erasmus University Rotterdam
Faculty of Economics, Econometric Institute
PO Box 1738, 3000 DR, Rotterdam, The Netherlands

Econometric Institute Report EI 2005-05

Abstract

The brand choice problem in marketing has recently been addressed with methods from computational intelligence such as neural networks. Another class of methods from computational intelligence, the so-called ensemble methods such as boosting and stacking have never been applied to the brand choice problem, as far as we know. Ensemble methods generate a number of models for the same problem using any base method and combine the outcomes of these different models. It is well known that in many cases the predictive performance of ensemble methods significantly exceeds the predictive performance of the their base methods. In this report we use boosting and stacking of neural networks and apply this to a scanner dataset that is a benchmark dataset in the marketing literature. Using these methods, we find a significant improvement in predictive performance on this dataset.

1. Introduction

A classical topic in marketing is modeling brand choice. This amounts to setting up a predictive model for a situation where a consumer or household, to purchase a specific product available in *k* brands, chooses one of these brands, given a number of household characteristics (such as income), product factors (such as price) and situational factors (such as whether or not the product is on display at purchase time). In the past, numerous different models have been proposed for brand choice problems. The most well known are the conditional and multinomial logit models (Franses & Paap, 2001; McFadden, 1973).

During the last decade, methods from computational intelligence such as neural networks have been proposed as an alternative to these classical models (Hruschka, 1993; West et.al., 1997). A recent contribution to the neural networks for brand choice literature is the article (Vroomen, Franses & van Nierop, 2004) in which neural networks are used to model a two-stage brand choice process: first a household chooses a so-called consideration set (i.e. a subset of the available brands which are most interesting for the

consumer at hand), and next the household selects a brand from this consideration set (Roberts & Lattin, 1997).

Another line of research which became very popular during the last decade both in the statistics and in the computational intelligence community, is the use of ensemble methods such as boosting, bagging and stacking (Hastie et al., 2001; Schwenk & Bengio, 2000; Tibshirani, Friedman & Hastie, 2000). These methods work by building not one model for a particular problem, but a whole series (ensemble) of models. These models are subsequently combined to give the final model that is to be worked with. The main advantage of these ensemble techniques is the sometimes spectacular increase in predictive performance that can be achieved. In marketing, ensemble methods are proposed in a forthcoming paper by two of the authors of this chapter (van Wezel & Potharst, 2004). Stacked neural networks for customer choice modeling was also applied in (Hu & Tsoukalas, 2003).

In this chapter we will explain some of these ensemble methods (especially boosting and stacking) and use them by combining the results of a series of neural networks for a specific brand choice problem. All methods explained will be demonstrated on an existing set of scanner data which has been extensively analysed in the marketing literature: the A.C. Nielsen household scanner panel data on purchases of liquid detergents in a Sioux Falls, South Dakota, market. This dataset contains 3055 purchases concerning 400 households of six different brands of liquid detergent: Tide, Wisk, Eraplus, Surf, Solo and All.

Summarizing, this chapter contains

- a brief description of classical models for brand choice
- a detailed description of how neural networks with one hidden layer may be used to model a brand choice problem, including
- a discussion of the features to be selected as explaining characteristics
- an explanation of how the concept of a consideration set can be modeled using a specific kind of hidden layer for the neural network
- an exposition on the ensemble methods of boosting and stacking, applied to the neural network models considered above
- demonstrations of all methods described using freely available real life scanner data.

2. Modeling brand choice: classical models

A classical problem in marketing is the so-called *brand choice* problem: trying to model the purchase behavior of a consumer given a number of explanatory variables. At purchase occasion i a consumer is faced with a choice between J brands 1, 2, …, J of a product he is going to buy. His final choice Y_i (which must be one of the brands 1,…, J) depends on three kinds of variables. The first kind of variable depends on the consumer or the purchase occasion, but not on the brand, for instance consumer income; we will denote such a variable by a capital X, so for instance the income of the consumer that purchases on occasion i is denoted by X_i. The second kind of variable depends only on

brand, not on purchase time and consumer. We will denote such a variable by a capital Z, so for instance the brand awareness of brand j is Z_j. The third group of variables depends both on consumer/purchase occasion and on brand. These variables are denoted by capital W, so for instance the price of the product of brand j that the consumer has to pay on purchase occasion i is W_{ij}.

Many models have been proposed for the brand choice problem in the marketing literature. In this section four representative models will be described. We start with the *multinomial logit* model. According to this model, $\Pr[Y_i = j \mid X_i]$, the probability that on occasion i a consumer chooses brand j, given the value of the expanatory variable X_i, has the following hypothesized functional form: for $j = 1, \ldots, J-1$ we have

$$\Pr[Y_i = j \mid X_i] = \frac{\exp(a_j + b_j X_i)}{1 + \sum_{j=1}^{J-1} \exp(a_j + b_j X_i)} \text{ and}$$

$$\Pr[Y_i = J \mid X_i] = \frac{1}{1 + \sum_{j=1}^{J-1} \exp(a_j + b_j X_i)}.$$

Note that, whatever the values of the parameters a_j and $ß_j$ and whatever the value of X_i, the sum of these probabilities over all brands equals one, as it should be. Note further that only variables of type X play a role in this type of model, variables of type W or Z are excluded. When purchase data is given, one may estimate the parameters a_j and $ß_j$ of this model by maximum likelihood methods.

The second kind of model we will mention is the *conditional logit* model, originally proposed by McFadden(1973). For this model the probability that brand j is chosen equals, for $j = 1, \ldots, J$

$$\Pr[Y_i = j \mid W_{i1}, \ldots, W_{iJ}] = \frac{\exp(a_j + bW_{ij})}{\sum_{j=1}^{J} \exp(a_j + bW_{ij})}.$$

Note again that the sum over all probabilities equals one, and in this type of model we have only variables of type W; here variables of type X and Z are excluded. Note also that parameter $ß$ does not have an index: the sensitivity for price-changes is supposed to be equal for all brands. This constraint may be relaxed using the model discussed next. Again, the parameters may be estimated using maximum likelihood.

The third model is the *general logit* model in which all three variable types (X, W and Z) may play a role. For this model the probability that brand j is chosen equals, for $j = 1, \ldots, J$

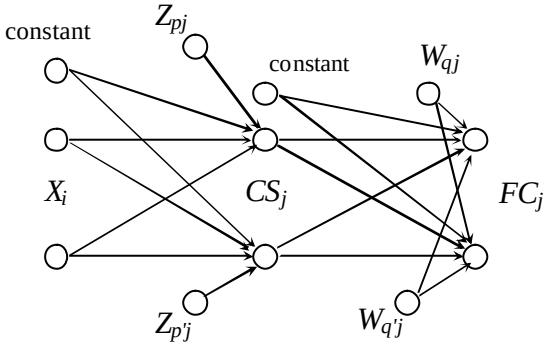
$$\Pr[Y_i = j \mid X_i, W_{i1}, \ldots, W_{iJ}, Z_1, \ldots, Z_J] = \frac{\exp(a_j + b_j X_i + g_j W_{ij} + dZ_j)}{\sum_{j=1}^{J} \exp(a_j + b_j X_i + g_j W_{ij} + dZ_j)}.$$

Note that not only all three variable types are incorporated in this model, but also variable W has a brand-specific coefficient, which relaxes the mentioned constraint of the conditional logit model.

3. Modeling brand choice: neural network models

In addition to the statistical models layed out in the previous section, brand choice can also be modeled using neural networks. This is a technique that was put forward at the end of the eighties in the machine learning community and its popularity soon became very high in very diverse fields, from linguistics to engineering, from marketing to medicine. For the brand choice problem neural networks have been proposed by several authors (Hruschka, 1993; Dasgupta et al., 1994; West et al., 1997; Hu et al., 1999; Hruschka et al., 2002; Vroomen et al., 2004). The neural network methodology is clearly explained in the first chapter of the introductory book (Smith & Gupta, 2002). This work also contains a number of other applications of neural network methodology to marketing problems, such as (Potharst et al., 2002).

We will introduce neural network modeling for the brand choice problem on the basis of a recently proposed neural network model that makes use of so-called *consideration sets* (Vroomen et al., 2004). Let us first review the theory on considerations sets. To this purpose, the process of choosing a product of a particular brand is viewed as consisting of two stages. In the first stage, the consumer reduces the set of available brands to a smaller subset: this subset is the consideration set, that will exclusively be considered when the consumer makes his final choice. In the second stage the consumer picks his final choice from the brands that reside in the consideration set (Roberts & Lattin, 1997). (Vroomen et. al., 2004) make use of a neural network with one hidden layer to model this two stage process. Their model can be visualized as follows:



As can be seen from this graph the network consists of three layers of nodes; roughly spoken, there is an input layer, followed by a hidden layer in which the consideration set (*CS*) is modeled, and an output layer that models the probability of the final choice for a brand (*FC*). Three types of input variables are used: household characteristics (X_i) like size of household or income level, brand characteristics (Z_{pj}) like price, promotion and advertising, and finally choice- and brandspecific characteristics (W_{qj}) like the observed price at the purchase occasion. Let the number of *X*-type variables be *I*, the number of *Z*-type variables *P* and the number of *W*-type variables *Q*. For each of the *J* brands there is a sigmoidal hidden node CS_j, that determines the probability that brand j is in the consideration set, as follows. For $j = 1,\ldots,J$

$$CS_j = G(a_{0j} + \sum_{i=1}^{I} a_{ij} X_i + \sum_{p=1}^{P} b_{pj} Z_{pj})$$

where the coefficients a_{0j}, a_{ij} and $ß_{pj}$ are parameters that must be estimated from the data and G is the logistic (or sigmoidal) function

$$G(x) = \frac{1}{1+e^{-x}}.$$

Next, in the second part of the network model, the probability of the final choice (FC_j) is determined using the consideration set, as follows. For $j = 1,\ldots, J$

$$FC_j = \frac{\exp(g_{0j} + \sum_{k=1}^{J} g_{kj} CS_k + \sum_{q=1}^{Q} d_{qj} W_{qj})}{\sum_{m=1}^{J} \exp(g_{0m} + \sum_{k=1}^{J} g_{km} CS_k + \sum_{q=1}^{Q} d_{qm} W_{qm})},$$

where again the coefficients $?_{0j}$, $?_{kj}$ and d_{qj} are parameters that must be estimated from the data. The final outcome of the neural network is the brand j that gets the largest probability FC_j. In (Vroomen et al., 2004) all these parameters are estimated (or, in the usual idiom the network is trained) by using the back-propagation algorithm, which is based on gradient descent.

4. The data set: scanner data for six liquid detergent brands

The data set we use to test our methods is the data set also used by (Vroomen et al., 2004) which is described by (Chintagunta and Prasad, 1998). This data set is freely available on the internet. It consists of scanner data on 3055 purchases of a liquid detergent of six possible brands: Tide, Wisk, Eraplus, Surf, Solo and All. These purchases concern 400 different households. For each purchase, the values of four different household-specific variables (X_i) are available ($I = 4$), furthermore there are four brand-specific variables (Z_{pj}, so $P = 4$) and three extra variables (W_{qj}, so $Q = 3$) Specifically, we have the following set of variables:

X_1 = the total volume the household purchased on the previous purchase occasion
X_2 = the total expenditure of the household on non-detergents
X_3 = the size of the household
X_4 = the inter-purchase time
Z_{1j} = the price of brand j in cents per ounce
Z_{2j} = 1 or 0 according to whether brand j was on feature promotion or not
Z_{3j} = 1 or 0 according to whether brand j was on display promotion or not
Z_{4j} = an indication of how recently brand j was purchased (between 0 and 1)
W_{1j} = 1 or 0 according to whether the household bought brand j on the previous purchase occasion, or not

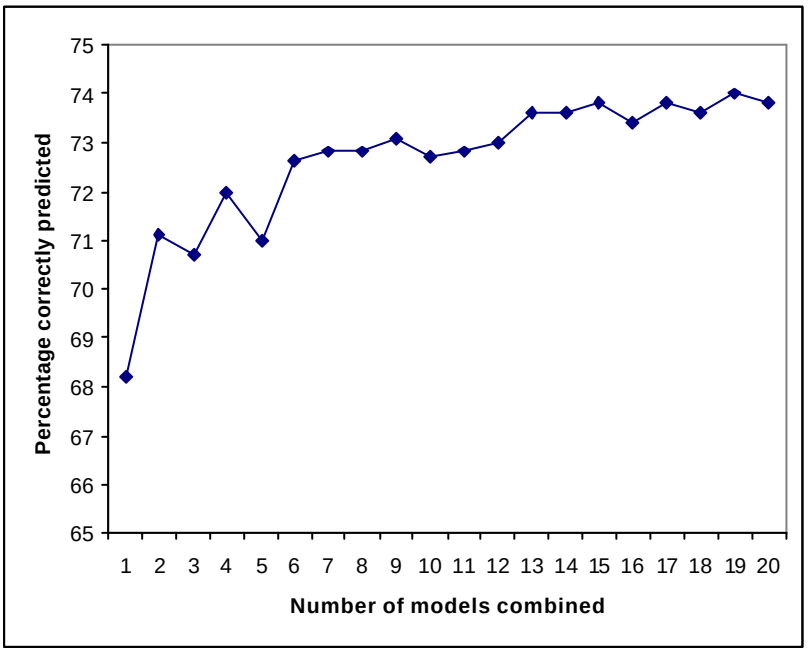
W_{2j} = 1 or 0 according to whether brand j was the brand most purchased over all previous purchase occasions, or not

W_{3j} = the fraction of recent purchases of brand j by the household divided by the total of all recent purchases of any detergent by the household

For a complete description of these variables, see (Vroomen et al., 2004). Because of incompleteness of some of the records we eliminated 798 of them, so we have 3055-798 = 2257 complete records available. All non-binary variables were scaled to the [0, 1]-interval in order to make all scales comparable.

5. Ensemble methods: bagging, boosting and stacking

In recent years there has been a growing interest in the datamining and statistics communities in so-called ensemble methods. These methods, also known as committee methods or opinion pools, work by combining different individual models (also called *base* models) for the the same problem. Great advantage of these methods is the gain in predictive performance that is often achieved by applying them. We illustrate this argument with the following example, based on a credit scoring dataset which is freely available on the internet. The problem which is addressed is to predict at the time of application for a loan whether the loan would ever be repaid or not. As can be seen from the graph below the error rate of such predictions drops from 32% when only one model is used to 25% when a combination of 20 models is used to predict the outcome.



Three of the most well-known ensemble methods are bagging, boosting and stacking, see (Hastie et al., 2001). *Bagging* is shorthand for 'bootstrap aggregating' and it works roughly as follows: from the dataset we draw a random sample[1] with replacement and build a model using only the data from this sample; a second model is built using only the data from a second random sample drawn from the original dataset, and so on. In this manner we arrive at a number of base models which are subsequently combined by casting votes. For instance, for an ensemble of 10 models, if for a certain input vector, 5 of the models predict brand 3, 2 models predict brand 1 and the remaining three models predict brand 2, the combined model predicts brand 3 (the majority of the votes). Since we will not use this method in this chapter we will not outline the exact algorithm for this method.

The second method we consider is *boosting.* The general idea of boosting is to create a sequence of models, where each model is trained on a reweighted version of the original dataset. Each training example in the dataset is assigned a weight and these weights are dynamically adjusted: when the model in a certain iteration of the algorithm makes an error in the classification of the n-th training pattern, the weight associated with this pattern is increased. This causes the model of the next iteration to focus on the patterns that were misclassified earlier. Continuing this way, an ensemble of models is created. The final model is a weighted majority vote of all the models from the ensemble.

The original boosting algorithm is called AdaBoost (Freund & Schapire, 1996), and we will now give an exact description of this algorithm. The original version of this algorithm is meant for two-class problems, where the classes are called –1 and +1. Since in our case we will not work with only two brands but with any number (J) of brands, we will have to adapt this algorithm later to our needs. Let us assume we have a dataset consisting of N data pairs (x_i, y_i), where x_i is a vector of input values and y_i the corresponding class value (either +1 or –1). In the description of the algorithm we will use the following notation: w_i is the weight of the i-th data pair, M is the number of boosting iterations (so the ensemble will consist of M models), ?(A) is an indicator function for boolean arguments A, which equals 1 if A is true and 0 if A is false. The function sgn(x) = 1 if $x \geq 0$ and -1 if $x < 0$. Note further that $F_m(x)$ can be any model based on the data, whether it be a statistical model, a neural network or a decision tree.

Here is the AdaBoost algorithm:

1. Initialize the boosting weights: for $i = 1, \ldots, N$

$$w_i = \frac{1}{N}$$

2. For $m = 1$ to M perform each of the following:

   (a) Train the base-model $F_m(x)$ on the dataset with weights $\{w_i : i = 1, \ldots, N\}$

   (b) Compute

[1] Of the same size as the original dataset!

$$err_m = \sum_{i=1}^{N} w_i c(y_i \neq F_m(x_i))$$

(c) Compute

$$a_m = \log(\frac{1-err_m}{err_m}) \quad (1)$$

(d) Redefine the weights w_i : for $i = 1, \ldots, N$

$$w_i = w_i \exp(a_m c(y_i \neq F_m(x_i)))$$

(e) Normalize the weights w_i : for $i = 1, \ldots, N$

$$w_i = \frac{w_i}{\sum_{k=1}^{N} w_k}$$

3. Output the final combined model:

$$O(x) = \text{sgn}(\sum_{m=1}^{M} a_m F_m(x)) \quad (2)$$

Let us take a look at this algorithm somewhat more closely to see how it works. First of all, in step 2(a), it is supposed that we are able to train a model on a dataset containing observations (purchases!) that each have a different weight. This can be implemented in several ways. We used the so-called resampling method which will be explained below. In step 2(b) the error rate err_m of the model built in the previous step is calculated. This is the in-sample training error rate, that allows for different weights in the dataset. Using this error rate, in step 2(c) the a_m coefficient is calculated. Provided the error rate < 0.5 (which it should be if the model does better than random guessing), this coefficient will be a positive number that increases if the error rate decreases. This a_m coefficient will be used in step 3 to weigh the votes from different models: the votes of models with lower error rates get a higher weight than those from models that perform less well. In step 2(d) the weights of the samples in the dataset are updated: for each sample that is classified incorrectly its weight is increased with the same factor, that involves the a_m coefficient. The weights of correctly classified samples remain unchanged. In step 2(e) the weights are normalized (they should sum to one). Finally, in step 3 the whole ensemble of models developed is combined using the weighted voting scheme as described.

The third ensemble method we consider is *stacking*. This term is usually referred to when there are two levels of learning involved: on the first level several models are trained on the dataset. These may be models of different types. Next, the models are combined not via a fixed voting scheme, but by learning an optimal combination method from the data. This is the second level of learning involved. Of course, such a stacking scheme can be implemented in many different ways. We will use a simple stacking scheme that is built on top of our boosting procedure: instead of combining the models from the boosting ensemble directly via Step 3, we will learn an optimal sequence of a_m coefficients from the data using the following algorithm:

1. Use the sequence of a_m coefficients that is constructed by the boosting algorithm as starting values
2. For iteration $k = 1$ to K perform the following steps:
   (a) for observation $i = 1$ to N do
      i. determine the output $O(x_i)$ using equation (2)
      ii. if $O(x_i) \neq y_i$ increase all coefficients a_m that belong to a model that gives the correct prediction y_i with a constant factor c: $a_m = c*a_m$.
   (b) determine the error rate of the new combined model (2) on the *validation set*; if it is lower than the best error rate so far, store this sequence of a_m's.
3. Output the final model (2) with the optimal sequence of a_m's.

Here K is the number of training iterations, and c is the constant update factor. (We will use $K = 50$ and $c = 1.01$.) The validation set is a dataset that is completely independent of our training set. How we construct such a validation set will be explained in section 6.

6. Comparing performance

Now that we have introduced the methods we are going to use and the brand choice dataset we will apply these methods to, we are in a position to knot things together. What we would like is make a comparison of the performance on the liquid detergent sales dataset of the methods we have introduced. Particularly, we would like to see the performance of Vroomen's model in comparison with boosting and stacking. As to performance we are especially interested in the predictive performance of the different models (since the claim is that it can be improved using ensemble methods). So, in this section we will be concerned with the predictive performance of the three methods we want to test.

In order to apply the boosting algorithm to the brand choice dataset there are a number of decisions to be made and problems to be solved. First of all, it has to be decided what kind of models we take as our base models, to serve as members of the ensemble. Since we want to make a comparison with Vroomen's model, it would be a good idea to use his model as our base model. Next, it should be decided what method we will use build a model on a dataset with weighted instances. We decided to use the so-called *resampling* method: we draw N times a random instance from the dataset (with replacement!) giving each instance i a weight w_i. Thus, we get a dataset of the same size as the original dataset, which contains however multiple copies of some instances with large weights while some instances with small weights might have disappeared from this new dataset. With this new dataset we build a model, which is thus based on a weighted dataset. One disadvantage of this method is that the generated model has a stochastic nature, since it is based on a random sample from the dataset. Another disadvantage is that some records just disappear from some of these randomly chosen datasets. However, this loss is fully compensated by the fact that we don't need to adapt the original model building method to the case of a dataset with weighted instances.

The most important adaptation we had to make regards the fact that we want our methods to work for any J brands, not just for two brands. If we call the brands we consider1,…, J, it follows that for the instance pairs (x_i, y_i) in our dataset we have $y_i \in \{1, \ldots, J\}$. Also, for each x, $F_m(x)$ (the output of the neural network) must be an integer in the range 1,…, J. With this situation in mind, it is not hard to see that the generalization of equation (2) to the situation of J brands should be replaced by the following:

$$O(x) = \arg\max_{1 \le j \le J} \sum_{m=1}^{M} a_m c(F_m(x) = j) \tag{3}$$

This equation expresses the procedure to cast a weighted vote for the brands among the M models (with weights a_m) and to pick the brand that got the largest number of (weighted) votes.

Another minor adaptation also had to be devised, regarding the situation of J instead of two brands. In equation (1), for the situation of two brands, we have seen that a positive coefficient is delivered provided the error rate err_m does not exceed 0.5. In the case of two brands this is an acceptable state of affairs, since random guessing between two brands results in an error rate of 0.5. So the only requirement that a model should meet, would be to be better than random guessing. However, in the situation of J brands, random guessing results in an error rate of $(J\text{-}1)/J$ since there are J-1 possibilities for an incorrect guess. Thus, if we set a comparable requirement to the formula for calculating the a_m coefficients as in the case of two brands, we should demand that a_m be positive as long as the error rate err_m does not exceed $(J\text{-}1)/J$. This is the case, if we define a_m according to the following equation:

$$a_m = \frac{1}{J-1} \log\left(\frac{1-err_m}{err_m} \cdot (J-1)\right) \tag{4}$$

The other important property, namely that a_m increases when the error rate err_m decreases remains true with this definition and it is equivalent to (1) in case $J = 2$. For these reasons, we used (4) to replace (1) in our boosting procedure.

Since we now have a complete description of the methods we used on the detergent dataset, we will now describe the experiments that we performed and their results. In order to get a fair view of the performance of a model, it should be tested on a completely independent test set. By the same token, a validation set, that is used to get an optimal sequence of a_m coefficents using the stacking algorithm, should be completely independent of both the training set and the test set. We used the following procedure to arrive at completely independent training, validation and test sets and an associated experimental cycle:

1. Split the 400 households randomly into a three groups, one of size 200 (training households, TR), and two groups each of size 100 (validation households and test households, VA and TE).
2. Training set = all purchases in the original dataset of 2257 records of the households from TR; validation set = all purchases of the households from VA; test set = all purchases of the households from TE.

3. Using the backpropagation algorithm on the training set a Vroomen model was created, and the accuracy of this model on the training set, the validation set and the test set was determined.
4. Using the boosting algorithm on the training set a whole ensemble of Vroomen models + a combined model was built; the accuracy of the combined model was determined for the training set, the validation set and the test set.
5. Using the stacking algorithm, starting from the ensemble of step 4 and making use of the validation set, an optimal sequence of a_m coefficents was trained for a new combined model; for this model the accuracy on training, validation and test set was determined.

This complete cycle was repeated ten times. The resulting mean accuracies over these ten runs, together with their standard deviations are displayed in the following table:

	Vroomen	Boosting	Stacking
training accuracy	80.7 ± 1.5	81.3 ± 1.9	81.8 ± 1.9
validation accuracy	76.4 ± 2.8	79.4 ± 2.3	79.3 ± 2.6
test accuracy	75.9 ± 1.8	78.6 ± 2.1	79.1 ± 2.5

We conclude from this table that the use of boosting and stacking results in a clear increase in performance, especially on validation data and completely independent test data. A stacked model predicts on average 3.2% better on unseen data than a Vroomen model. Actually, both boosting and stacking perform better than Vroomen on the test set in all of our ten runs. With boosting the increase varies from 0.6% to 4.5% and with stacking from 0.7% tot 5.5%. So indeed, the expected increase in predictive performance is confirmed on the detergent dataset.

Another remarkable outcome of these experiments is that in our experiments the Vroomen model does better than reported in (Vroomen et al., 2003) on the same dataset: they report an accuracy of 73.9% on an independent test set, whereas we get 75.9% on average. One reason for this difference might be that we use three extra variables in the second stage of the model (the W_{qj} variables) while they use only one extra variable, namely price. Another difference is that we repeat the whole experiment ten times, whereas they reported only one experiment. Their random split of the dataset into training, validation and test set might just have been an unlucky one. By repeating the experiment ten times we get a fair idea of the stability of our results on this dataset.

7. Trends and conclusions

In this chapter we showed how some new methods from the field of computational intelligence (ensemble methods such as boosting and stacking) could be used for a traditional marketing problem like brand choice. We found that indeed the predictive performance of the models based on the ensemble techniques improved even compared to the most sophisticated existing model that we found in the literature. Although predictive performance is one aspect that candidate models should be judged with, there are more

aspects that should be taken into consideration. One aspect that has not been considered here is the interpretability of the generated model. Since models built by ensemble methods consist of a combination of different (sometimes many!) base models the complexity of the final model is usually high, making it difficult to interprete. This is one of the themes that should be taken into account by future serious work into this direction.

References

Chintagunta, P.K. & Prasad, A.R. (1998). An empirical investigation of the dynamic McFadden model of purchase timing and brand choice: implications for market structure. *Journal of Business and Economic Statistics, 16*, 2-11.

Dasgupta, C.G., Dispensa, G.S., Ghose, S. (1994). Comparing the predictive performance of a neural network model with some traditional market response models. *International Journal of Forecasting, 10,* 235-244.

Franses, P.H. & Paap, R. (2001). *Quantitative Models in Marketing Research.* Cambridge, UK: Cambridge University Press.

Freund, Y. & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*(148-156). San Francisco, CA, Morgan Kaufmann Publishers.

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The elements of statistical learning.* New York, Springer Verlag.

Hruschka, H. (1993). Determining market response functions by neural network modeling: a comparison to econometric techniques. *European Journal of Operational Research, 66,* 27-35.

Hruschka, H., Fettes, W., Probst, M. & Mies, C. (2002). A flexible brand choice model based on neural net methodology. A comparison to the linear utility multinomial logit model and its latent class extension. *OR Spectrum, 24,* 127-143.

Hu, M.Y., Shanker, M., Hung, M.S., (1999). Estimation of posterior probabilities of consumer situational choices with neural network classifiers. *International Journal of Research in Marketing, 16,* 307-317.

Hu, M.Y. & Tsoukalas, C. (2003). Explaining consumer choice through neural networks: The stacked generalization approach. *European Journal of Operational Research, 146*, 650-660.

McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior. In P. Zarembka (Ed.), *Frontiers in Econometrics* (105-142). New York: Academic Press.

Potharst, R., Kaymak, U. & Pijls, W. (2002). Neural networks for target selection in direct marketing. In Smith, K. & Gupta, J. (Eds.), *Neural Networks in Business: Techniques and Applications* (89-110). Hershey PA, Idea Group Publishing.

Roberts, J.H. & Lattin, J.M. (1997). Consideration: Review of research and prospects for future insights. *Journal of Marketing Research*, *34*, 406-410.

Schwenk, H. & Bengio, Y. (2000). Boosting neural networks. *Neural Computation, 12*, 1869-1887.

Smith, K. & Gupta, J. (Eds.) (2002), *Neural Networks in Business: Techniques and Applications*. Hershey PA, Idea Group Publishing.

Tibshirani, R., Friedman, J. & Hastie, T. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics, 28,* 337-407.

Vroomen, B., Franses, P.H. & van Nierop, E., (2004). Modeling consideration sets and brand choice using artificial neural networks. *European Journal of Operational Research, 154*, 206-217.

West, P.M., Brockett, P.L. & Golden, L.L. (1997). A comparative analysis of neural networks and statistical methods for predicting consumer choice. *Marketing Science, 16* (4), 370-391.

Wezel, M. van, & Potharst, R. (2004) Improved customer choice predictions using ensemble methods. Technical Report from http://www.few.eur.nl/few/people/mvanwezel/.