HSC Research Report

HSC/95/02

Analysis of ROBECO data by neural networks

Wojtek Kowalczyk*
Rafał Weron*,**

* Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands
** Institute of Mathematics, Wrocław University of Technology, Poland

Hugo Steinhaus Center
Wrocław University of Technology
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
http://www.im.pwr.wroc.pl/~hugo/

Analysis of ROBECO data by Neural Networks

Wojtek Kowalczyk

and

Rafal Weron[1]

Department of Mathematics and Computer Science
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam

07.07.95

Introduction

Our task was to find a model for classifying **Robeco** clients into four classes according to their degree of satisfaction. Each client was represented by a vector of 30 variables, which could be split into two groups: variables related to the specific client (*jarenklant*, *rendement*, etc.) and socio-geographical variables (*social class*, *wealth*, *education*, etc.) characterizing the area in which the client lived. The original set contained 21, 90, 288, and 146 vectors from group 1, 2, 3, and 4 respectively. Additionally, an independed *validation* set was provided with 3, 12, 48 and 32 vectors from corresponding groups.

The process of training neural networks requires each class to contain about the same number of vectors. Therefore vectors in groups 1, 2 and 4 were multiplied appropriately. To avoid overtraining the modified set was split into *training* (1029 vectors) and *test* (115 vectors) sets. However, this procedure had a serious impact on the performance of the network. Vectors from group 1 had to be multiplied 14 times. During the training phase the network memorized them instead of generalizing and finding some common factors. Moreover, 48 out of 95 vectors in the validation set belonged to group 3. A classification rate of about 50% for this set could be reached by a network classifying all vectors into group 3.

[1] On leave from Institute of Mathematics, Wrocğaw University of Technology, WyspiaŒskiego 27, 50-370 Wrocğaw, Poland.

Data Analysis

We approached the problem via a number of different types of neural networks. It turned out that *Back-Propagation* and *Learning Vector Quantization* (LVQ) networks were most suitable (see Appendix for more details about these networks). We decided to work with the latter, because they performed slightly better in terms of training speed and accuracy. After some trial runs we found what seemed to be the best performing LVQ network. It had 30 input units, 24 units in the hidden Kohonen layer, and 4 output units - each representing one group. The *conscience factor* was set to 0.5. The first learning phase (LVQ1) consisted of 50000 iterations, the second one (LVQ2) of 30000 iterations. The following tables show test results on the three sets.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	239		13	12	***264***
2.00	6	195	24	18	***243***
3.00	7	22	204	26	***259***
4.00	7	13	30	213	***263***

Table 1. Confusion matrix on the training set. Classification rate is 82.7%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	27		1	2	***30***
2.00	9	6	3	9	***27***
3.00	3	8	13	5	***29***
4.00	1	3	6	19	***29***

Table 2. Confusion matrix on the test set. Classification rate is 56.5%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	1		1	1	***3***
2.00		3	3	6	***12***
3.00	1	14	24	9	***48***
4.00	1	7	11	13	***32***

Table 3. Confusion matrix on the validation set. Classification rate is 43.2%.

As a second attempt we trained the same LVQ network using the *savebest* option. During the training stage, once every 1000 iterations the network was evaluated on the test set. The best performance was reached after about 90000 iterations of the LVQ1 phase. The following tables show test results on the three sets.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	239		13	12	***264***
2.00	6	198	21	18	***243***
3.00	8	19	210	22	***259***
4.00	7	10	24	222	***263***

Table 4. Confusion matrix on the training set. Classification rate is 84.5%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	27		1	2	***30***
2.00	6	9	3	9	***27***
3.00	4	7	11	7	***29***
4.00	1	2	4	22	***29***

Table 5. Confusion matrix on the test set. Classification rate is 60.0%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	1		1	1	***3***
2.00		3	3	6	***12***
3.00	1	12	25	10	***48***
4.00	1	8	9	14	***32***

Table 6. Confusion matrix on the validation set. Classification rate is 45.3%.

Up to this point the performance, especially generalization abilities, of our neural network model was disappointing. We thought that it could do better if we used only the most important (most influential according to **Robeco** experts) variables: *jarenklant* and *rendement*. Moreover, we crossed out vectors representing clients, who had only savings accounts. The following tables show the test results for an LVQ network with 7 input units, 24 units in the hidden Kohonen layer, and 4 output units. The *conscience factor* was set to 0.5. The first learning phase (LVQ1) consisted of 50000 iterations, the second one (LVQ2) of 30000 iterations.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	176	13	13	12	***214***
2.00	3	168	39	18	***228***
3.00	16	41	126	36	***219***
4.00	17	34	47	93	***191***

Table 7. Confusion matrix on the training set. Classification rate is 66.0%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	20	1	1	2	***24***
2.00	6	9	9	3	***27***
3.00	3	8	6	5	***22***
4.00	1	2	7	11	***21***

Table 8. Confusion matrix on the test set. Classification rate is 48.9%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00			1	1	***2***
2.00	2	5	5		***12***
3.00	4	14	12	11	***41***
4.00	2	8	6	7	***23***

Table 9. Confusion matrix on the validation set. Classification rate is 30.8%.

These results were even more disappointing. We thought it might be interesting to train the network without the most ‘important’ variables. Following tables present test results for data sets including only socio-geographical variables and without vectors representing clients, who had only savings accounts. We used an LVQ network with 14 input units, 24 units in the hidden Kohonen layer, and 4 output units. The *conscience factor* was set to 0.5.

The first learning phase (LVQ1) consisted of 50000 iterations, the second one (LVQ2) of 30000 iterations. Surprisingly it performed a little better than in the previous experiment.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	201		13		***214***
2.00	9	168	33	18	***228***
3.00	11	24	162	22	***219***
4.00	12	20	32	127	***191***

Table 10. Confusion matrix on the training set. Classification rate is 77.2%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	23		1		***24***
2.00		3	12	12	***27***
3.00	3	6	8	5	***22***
4.00	2	2	4	13	***21***

Table 11. Confusion matrix on the test set. Classification rate is 50.0%.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00			1	1	***2***
2.00		2	6	4	***12***
3.00		6	22	13	***41***
4.00	1	8	11	3	***23***

Table 12. Confusion matrix on the validation set. Classification rate is 34.6%.

Finally, to get a better insight into some global properties of our models we ran two experiments. In the first one we've simply applied our best model to the original training set (without duplicated vectors). The resulting confusion matrix (table 13) allowed us to compute the average satisfaction score for each satisfaction group. These averages were 2.0, 2.4, 3.0 and 3.7 for groups 1, 2, 3 and 4, respectively. As might be expected, the average for group 1 is bigger than 1.0 (all cases must get score not smaller than 1.0, so the average must be bigger than 1.0) and similarily, the average for group 4 is smaller than 4.0.

Predicted:	*1.00*	*2.00*	*3.00*	*4.00*	***Total***
1.00	19		1	1	***21***
2.00	4	69	8	9	***90***
3.00	12	26	221	29	***288***
4.00	4	6	14	122	***146***

Table 13. Confusion matrix on the training set. Classification rate is 79.1%.

In the second experiment the sensitivity analysis of the best model have been performed. We measured the influence of modifications of input variables on output. Because we have 30 input variables and 4 output units, the result of this analysis is a matrix 30x4 (Table 14).

GROUP	**1**	**2**	**3**	**4**	**Importance**
JARENKLA	-1.74	-2.36	4.85	-0.75	9.70
WELSTAND	4.03	0.84	-5.12	0.25	10.23
GEZINSFA	6.55	0.53	-2.06	-5.02	14.16
EIGENDOM	0.33	0.56	-0.97	0.08	1.95
HUURPRIJ	4.32	-2.00	-0.41	-1.90	8.63
KOOPPRIJ	0.42	6.62	-3.83	-3.21	14.07
SOCKLASA	3.88	-3.66	-2.58	2.35	12.47
SOCKLAB1	5.84	-3.52	-3.33	1.01	13.70
SOCKLAB2	**-9.18**	**0.52**	**1.69**	**6.97**	**18.35**
SOCKLASC	-5.13	-1.88	-0.15	7.17	14.34
SOCKLASD	-1.00	1.71	-0.73	0.01	3.45
OPLEID_L	-1.71	7.40	-2.45	-3.24	14.80
OPLEID_M	2.54	-3.85	-0.18	1.49	8.08
OPLEID_H	**-5.31**	**7.88**	**1.59**	**-4.16**	**18.95**
BELEGGEN	1.96	-5.72	-0.92	4.68	13.27
H_010694	-0.41	-3.95	4.16	0.20	8.72
H_250594	-0.46	-4.32	5.94	-1.15	11.87
H_010594	**-2.60**	**-2.97**	**8.25**	**-2.68**	**16.51**
H_010394	-3.53	-0.01	2.62	0.92	7.09
H_011293	1.32	-1.97	-1.27	1.93	6.49
H_010693	-1.46	1.94	4.13	-4.61	12.13
H_010392	-0.31	-3.23	-2.42	5.96	11.91
R_250594	0.33	-5.07	3.08	1.66	10.15
R_010594	0.44	1.41	-3.36	1.51	6.73
R_010394	3.80	-0.03	-2.68	-1.09	7.60
R_011293	**-3.99**	**-4.00**	**5.28**	**2.71**	**15.98**
R_010693	1.05	-0.07	2.48	-3.46	7.06
R_010392	3.06	0.14	3.70	-6.90	13.79
LEEFTIJD	-4.58	5.40	-1.31	0.49	11.77
GESLACHT	**-4.75**	**-3.02**	**4.53**	**3.24**	**15.55**

Table 14. The results of sensitivity analysis.

Each row indicates how increasing value of the corresponding variable contributes to falling into a given satisfaction group. For example, numbers associated with the variable JARENKLANT, -1.74, -2.36, 4.85 and -0.75 indicate that increasing this variable should lead to clients that fall into group 3. Because groups are disjoint, the sum of influences is always 0. Nevertheless the importance of a given variable may be measured by calculating the sum of absolute values of influences (for JARENKLANT it gives 9.70). This 'global' importance can be found in the last column of Table 14. Variables with importance higher than 10% are in boldface. By analysing colums of our tables one can find variables that are most characteristic for given satisfaction group. For example, if we take a look into the first satisfaction group we will see, that variables GEZINSFA and SOCLAB1 have a big positive influnece (6.55% and 5.84%, resp.), whereas SOCKLAB2, SOCKLASC and OPLEID_H have negative influence (-9.18%, -5.13% and -5.31%, resp.).

Conclusions

In all three experiments performance fell well below our expectations. The best performance (classification rate of 45.3% on the validation set) was achieved on a training set in which all 30 variables were used; 34.6% was attained on a training set with only socio-geographical variables; the worst performance (30.8%) was achieved when only JARENKLANT and rendement variables were taken into account. There might be at least 3 reasons for such weak performance of our models:

1. ***There is no significant dependency between input variables and satisfaction.***
Our third experiment (with variables limited only to socio-geographical data), where a good model (on the training set) could be constructed, seems to confirm this hypothesis.

2. ***The number of cases in the data set is too small***.
Due to a relatively big number of variables (compared to the number of cases) the resulting network must have relatively many connections. This may lead to immediate overfitting (the network 'memorizes' single vectors instead of 'learning' most significant features).

3. ***The number of 'unsatisfied clients' in the dataset is too small***.
In our dataset the number of cases in group 1 (satisfaction = 1) was 21, which is very little, when compared to the number of all cases (545). It is not surprising, that the network couldn't learn some general properties of this group.

Appendix. Learning Vector Quantization Networks.

LVQ networks are classification networks. It means that they try to find an optimal (with respect to the given classification task) clustering of the data set. A typical LVQ network has three layers: input layer, Kohonen layer and output layer. In our case we used a network with 24 units in the Kohonen layer (6 units per satisfaction group) and 4 output units (one per satisfaction group). Our network tried to split the training set into 24 small clusters in such a way that each satisfaction group could be represented by a union of 6 of these clusters. The clustering algorithm tries to preserve topological properties of input vectors (i.e. the distance from the center of the given cluster decides whether a vector falls into this cluster or not). Moreover, a special mechanism, *conscience factor*, takes care that all clusters have more or less the same size. The biggest advantage of LVQ networks over backpropagation networks is the speed with which they learn (very quickly) and very transparent model that they provide (small clusters that may have simple interpretations). On the other hand, LVQ networks, in contrast to backpropagation networks are not suitable for function approximation or forcasting.