Understanding Customer Choice Processes Using Neural Networks

Walter A. Kosters, Han La Poutré and Michiel C. van Wezel

Leiden University, Department of Computer Science
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Phone: +31 71 527 7059; Fax: +31 71 527 6985
Email: {kosters,han,michiel}@wi.leidenuniv.nl

Abstract

We propose and examine two different models for customer choices in a wholesale department, given the actual sales. Both customers and products are modelled by points in a k-dimensional real space. Two possible strategies are discussed: one buys the nearest option from categories of products, the other buys all products within a certain radius of the customer. Now we deal with the following problem: given only the sales list, how can we find again the original coordinates corresponding with customers and products. In particular we are interested in the dimension of the space. We are looking for low dimensional solutions with small errors compared to a real sales list. Theoretical complexity of these problems is addressed: they are very hard to solve exactly; in fact they are NP-complete. We use techniques from neural networks for both artificial and real life data, and obtain promising results.

Appeared as pp. 167–178 in:
H.F. Arner Jr., editor, Proceedings of the First International Conference on the Practical Application of Knowledge Discovery and Data Mining, PADD'97, London, April 1997.

1 Introduction

In this paper we examine large data files concerning sales in a wholesale department. Every customer buys many items; he or she may choose from an extensive catalogue of different products. At the end of every period only a list of all sales remains. If a customer places a new order, he or she is considered to be a new customer: the identity of the customer is hidden in this list.

We propose different models to understand the behaviour of the customers. In both models customers and products are modelled as points in a k-dimensional real space. These k dimensions are supposed to represent aspects that influence consumer behaviour, such as price, product quality, appearance and so on. Without using any interpretation of these dimensions, we try to find the number of relevant dimensions and the coordinates of customers and products. Contrary to many approaches (cf. [1]) our methods use only very partial information; instead of direct attributes like age or price, we only use the sales list.

In the first model, following [5], the products are grouped into a small number of categories. Every customer has to buy a product from each category, choosing from the options (products) available in it. The customer chooses —from the different options within each category— the nearest one. In the second model every customer buys exactly those products that are within a certain distance, the so-called radius of the customer. Here some notion of distance is necessary. We examined the Euclidean distance and some Manhattan-like distances.

Now the problem can be easily described. Given a list of sales, determine a dimension k such that the model generates those same sales — up to a certain error. Here k should be as small as possible. Furthermore, we also want to find the coordinates of both customers and products.

We will prove that for the first model this problem is already NP-complete (see [2]) in special cases. This means that an exact solution is beyond reach, possibly for many years to come. This theoretical complexity justifies the use of approximating algorithms, such as neural networks, see [4].

Next we provide some results of experiments on artificial data, using neural networks. Customer and product points in k-space were generated randomly, and sales were generated according to one of the models. Throwing away the original points, the problem was to relocate their coordinates. Experiments show that this is feasible, especially in the case where every customer buys lots of products. The neural networks involved are related to simple competitive neural networks.

Real data offer more difficulties. Motivated by the results on artificial data we restricted the experiments to those customers who bought at least 40 products. At first it seemed hard to improve upon the so-called naive error: the system that buys nothing at all has a relatively small error. Fortunately we had several runs that were promising. The experimental results can be analysed by e.g. cluster analysis in order to obtain a better understanding of the coordinates — and the dimensions.

We would like to thank Camiel van Breugel, Marc Zegveld, and the anonymous referees, who all contributed —in quite different ways— to this paper.

2 Description of the Problem and the Models

Suppose we are given n customers and m different products. We assume that customers buy each product at most once. We have to analyse the so-called sales list: for every customer we know exactly the products he or she bought. We now propose two models to understand the behaviour of the customers.

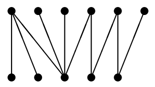
- First we describe the *option model*, cf. [5]. The products can be divided into c disjoint categories $O_1, O_2, \ldots, O_c$. In each category we have a number of products, say $o_i \geq 2$ in category O_i $(i = 1, 2, \ldots, c)$; these products are called the options or the alternatives. Every customer has to choose *exactly one* of the options in each category. We suppose that the sales list has this property. If customers and options are points in a k-dimensional real space, the model lets the customer choose the nearest option. We assume that there always is exactly one option that is the nearest; if not, we slightly disturb the customers' coordinates. Here we need a certain metric on the space. It is also possible —but not obvious— to deal with the concept of non-buying in this model: an extra option could be added, corresponding with the situation where the customer does not want to choose from the original options. Finally, the division into categories is in principle done by the wholesale department.
- Secondly, we describe the *product model*. Now each customer j also has a *radius* r_j $(j = 1, 2, \ldots, n)$, proportional to the number of products he or she bought. If customers and options are points in a k-dimensional real space, the model lets the customer buy precisely the products within its own radius.

Now the problem is the following. Given real data, i.e. customers with their sales, is it possible to find points in k-space such that either the option model or the product model predicts these sales within some reasonable error bound? Here the dimension k should be as low as possible. In fact, it can be shown that if $k = m$ a zero error solution can be given. To do so for the option model, take the vector v_i $(i = 1, 2, \ldots, m)$, corresponding with product i, to be $(0, \ldots, 0, 1, 0, \ldots, 0)$ (a one in coordinate i), and just add the products bought in reality to produce the vector corresponding with customer j $(j = 1, 2, \ldots, n)$. However, in order to hope for some real life interpretation we are interested in situations where k is in the order of magnitude 10.
In the previous paragraphs an error is mentioned. This error may be defined —for the moment— as the number of products/options the model buys, but the customers do not, added to the number of products/options the customers buy, but the model does not. So the naive error, where the system buys nothing at all, equals the total number of sales.

3 Complexity of the Models

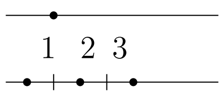
In this section we shall examine a special case of the option model, and we shall prove that the corresponding decision problem is NP-complete in the sense of [2]. These problems are very difficult indeed; up to this moment nobody has been able to find efficient solutions to any of these problems. Often approximating or probabilistic algorithms are used, for instance neural networks. Since exact solutions are not feasible, one would be happy to accept near optimal solutions. A famous example is the Traveling Salesman Problem, where a person is asked to visit a given number of cities using a route as short as possible. NP-complete problems are usually formulated as decision problems with a yes/no answer. The most common technique used to show NP-completeness is called reduction, where a known NP-complete problem is reduced —in a very precise way— to the problem at hand. For instance, in [2] a reduction is given from the Traveling Salesman Problem to the problem of finding a Hamiltonian circuit in a graph: a closed route connecting all nodes, visiting them once.

We first describe the special case of the option model we would like to address. We have n customers, who must choose from only $c = 2$ categories, with a total of m products. We identified customers who had bought exactly the same products. The customer choices are easily represented using a graph consisting of two horizontal rows of nodes, where the nodes in the first row correspond with the options from the first category, and the nodes in the second row with the options from the second category. The edges now correspond with the choices made: every customer corresponds with a unique edge. So the graph has m vertices and n edges. We call this graph a *sales graph.* We may assume that this graph is connected. An example, with $m = 6 + 5 = 11$ and $n = 10$:

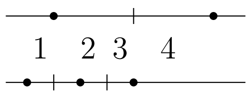


This graph is *bipartite*: the nodes can be split into two nonempty disjoint sets, such that every edge is incident with a node from both sets. The converse also holds: every bipartite graph can be viewed as a sales graph.

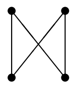
Now we return to our original problem, and we consider the special case of dimension $k = 1$: "we only look at the price of the products". Is it possible to find an exact solution in this case? In other words, is it possible to attach real numbers to customers and options (where customers choose the nearest option), in such a way that the given graph represents the corresponding sales? In dimension 1 we should divide the real axis for both categories into disjoint intervals; in higher dimensions we get Voronoi cells here.
We now explain a way to find suitable coordinates for customers and products. We proceed from left to right, adding one customer at a time. Notice that every new customer adds a new option too. So suppose coordinates for l customers and the corresponding options —coming from the left— have been chosen. We now choose a new option coordinate to the right of the previous one (in the same category) in such a way that the latest customer coordinate is still correct. In the example above, suppose we have correctly chosen $l = 3$ customers, say as:



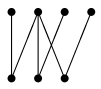
Here the real axis is depicted three times: the first horizontal line shows the first category, the second horizontal line represents the second category; in the middle we see numbers representing the customers. The dots represent the coordinates for the options. The small vertical lines denote the boundaries of the intervals, precisely in the middle between two option coordinates. For instance, customer 2 buys the only option available from the first category and the second option from the second category. Now we want to add the fourth customer, who bought a new option from the first category, and bought the rightmost option from the second category. We have to take care that we do not disturb the third customer, so the small vertical line in the upper row has to be to the right of the 3. We get:



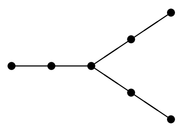
Continuing in this way we see that horizontally stretching the sales graph we arrive at the desired coordinates. If we shrink all intervals by the same factor, we can embed all coordinates in the interval $[0, 1]$.
When does this construction fail? It is easily seen that this happens in the following situation:



In this case we have to cut the real axis into two parts —for both categories— in such a way that all four combinations have a nonempty intersection. This is impossible. In every way we choose the coordinates for customers and options, at least one customer will be treated incorrectly. In fact, if for the first category, with options A and B, we choose real numbers a and b with $a < b$, and for the second category, with options C and D, we choose real numbers c and d with $c < d$, then a customer corresponding to x chooses A if $x < (a + b)/2$ and B if $x > (a + b)/2$ and similarly for C and D. Now —given real a, b, c and d— it is not possible to find x_1 that buys both A and D, and x_2 that buys both B and C.
In general we see that cycles are forbidden, in other words: the sales graph should be acyclic, i.e. not contain any paths from a node to itself. It seems that in the example it could be possible to deal with this problem by allowing nondeterminism, for instance with x's precisely in the middle of a and b. However, first of all the model would become more complex, and secondly this would also fail in the case of cycles with more than four nodes.
Furthermore the construction also fails for:

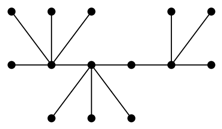


However hard one tries, it is impossible to arrange the option nodes in such a way that crossing edges can be removed. Since crossing edges correspond to the fact that customers are treated correctly or not, they should be avoided. We can conclude that a sales graph can be realised in the way we want, if and only if it contains no cycles and no subgraphs isomorphic to graph H:



Here we replaced the sales graph from the previous example by an isomorphic one. The minimal number of edges that should be omitted in order to obtain a graph that meets the conditions, is exactly the minimal number of customers that is treated incorrectly.

So we are motivated to examine the following decision problem, called SG (Sales Graph). Given a bipartite graph G. Does G possess a spanning tree (a connected subgraph without cycles, containing every node) that does not have a subgraph isomorphic to H? Stated otherwise, does G have a spanning tree that looks like:



So we should have a long "spine", to which several degree one nodes are attached.
We now arrive at the main result of this section:

Theorem SG *is* NP-*complete.*

The proof proceeds as follows. At first we notice that SG is in the class NP. Given a candidate subgraph, it is easy to verify the conditions. Next we will reduce a known NP-complete problem to this one. We choose problem GT39 from [2]. Given a bipartite graph G; does G possess a Hamiltonian path, i.e. a path connecting all nodes without repeated visits to nodes?
The reduction goes as follows. Suppose that a bipartite graph G is given; we would like to know whether or not G admits a Hamiltonian path. We construct a bipartite graph G' from G by adding a node x' to every node x of G, only connected to the rest of G' by an edge incident with x and x'.
Now we notice that G' has a spanning tree, without subgraphs isomorphic to H, if and only if G has a Hamiltonian path. In fact, suppose G has a Hamiltonian path; attach the extra nodes to this path, and we are done. If the desired spanning tree of G' exists, all nodes x' occur as degree one nodes (by construction they are incident with precisely one edge); omit these nodes in order to obtain a Hamiltonian path for G. The remaining nodes indeed form a path, since if there were a node of degree at least three, we would have a subgraph isomorphic to H in the original G'.

Now that we have shown that SG is NP-complete, we can infer that given customers and their choices, it is not feasible to determine the minimal error that can be reached. If this were possible, the existence of a spanning tree without subgraphs isomorphic to H would be immanent. It is therefore appropriate to turn to approximating algorithms, such as neural networks.

4 Description of the Neural Network

The neural network we used in our experiments is inspired by simple competitive neural networks (see e.g. [3] or [4]). In such neural networks, the weight vectors associated with the neurons are merely points in a k-dimensional real space if the input data for the network are k-dimensional real vectors. Each time a data vector is presented to a competitive neural network, a "winning unit" w is determined by calculating which unit has its weight vector closest to the presented pattern (in a Euclidean sense). Next, this units' weight vector is moved towards the presented pattern. This increases the chance of unit w being the winner if the same input pattern is presented again later on.
In our case, where we analyse sales lists, all customers and all products (or options within a category) are represented by a separate competitive unit. The weight-vectors of the units correspond with the points in a k-dimensional real space describing the customers and products (or options).
Of course, the learning process has to be modified in order to be able to work with sales lists. The aim of the learning process is to bring weight vectors of each customer and weight vectors of the products he or she has bought sufficiently close to each other,

whereas weight vectors of the products the customer has not bought should be at a sufficient distance from each other.
The meaning of the word sufficient is crucial here, and it is determined by the model used. In the case where a customer buys everything within a certain radius of his own weight vector, sufficiently close means within this radius, and sufficiently distant means outside this radius. In the case where a customer buys the closest one of a number of options within a category, sufficiently close means closer than all the other options in the category, and sufficiently distant means at a greater distance than the option the customer *did* buy from the category.
Now, the training of our network roughly proceeds as follows. First, the network weights are randomly initialized in the $[0:1]$ hypercube. Next, the network is trained for a number of iterations. In each iteration, a number (say 5000) of random customer/product pairs are examined. If the customer and the product are not sufficiently close, the weight vectors representing them are pulled towards each other in the k-dimensional hypercube. If they are not sufficiently far apart, they are pushed away from each other. At the end of an iteration the coordinates are renormalised in order to make optimal use of the full unit cube. Iterating is stopped if the error (which is described below) has not decreased anymore for a sufficiently large number of iterations.
The error associated with this neural network can be divided into two types:

only reality bought error: this error occurs when a product is *not* sufficiently close to a customer, so the product is not bought according to our model, but *is* bought according to the real data;

only model bought error: this error occurs when a product is not sufficiently distant from a customer, so the product *is* bought according to our model, but it is *not* bought according to the real data.

At the end of each run we in particular have four significant figures:

"both bought": number of products both reality and the model bought;

"only reality bought": number of products only reality bought;

"only model bought": number of products only the model bought;

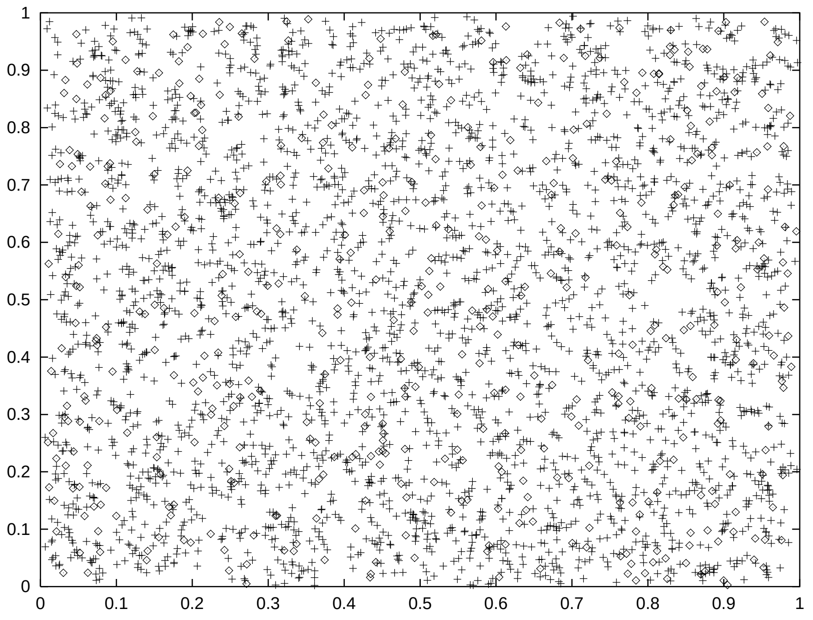
"none bought": number of products none bought.

In most experiments this last number is enormous. If we talk of the number of products here, we mean the number of combinations customer-product.
The error we make could be defined as the sum of "only reality bought" and "only model bought", perhaps relative to "both bought" or the total number of sales. If we think of the naive error, where the model buys nothing at all, it is more illustrative to show all numbers involved. Notice however that the sum of all four always equals the number of customers times the number of products, and that the sum of "both bought" and "only reality bought" always equals the total number of sales.

5 Results on Artificial Data

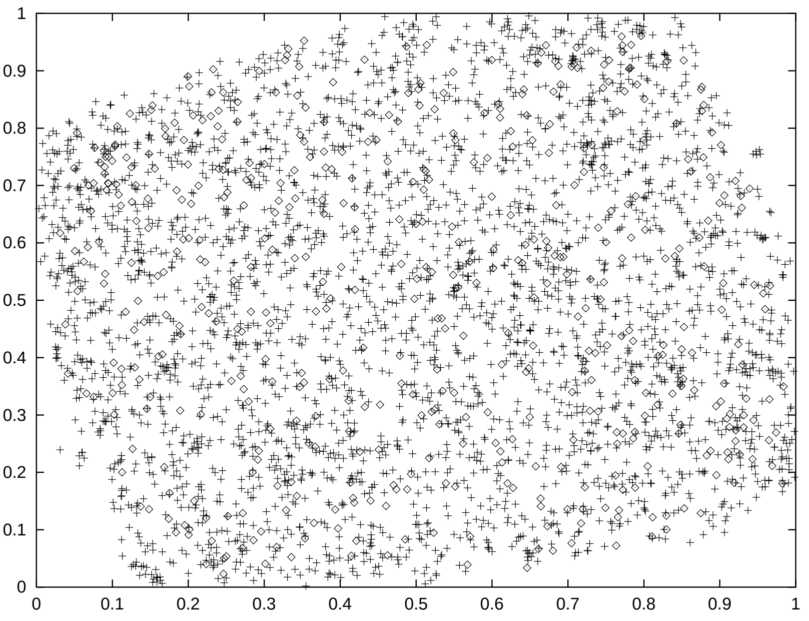
Suppose that our models resemble reality. Since we only get the sales lists, it still is difficult to find the coordinates of all customers and products involved. Even the dimension is hard to find. Therefore we have to develop methods to examine sales lists. In order to show that these methods work, we apply them to artificial data first. This method seems promising, see [5] for an application in the case of the option model. The data are generated by randomly assigning coordinates to customers and products, and producing the corresponding sales list using the model. In this section we shall describe several experiments. We shall also draw some conclusions for the real situation.
Let us now first restrict our attention to dimension $k = 2$ in the product model. In this case results can be easily visualised. We chose $n = 500$ customers and $m = 3000$ products, so experiments took little time. As a good run we present a situation where 116638 products were sold, whereas the total error was a mere 8928, with 7122 "only model bought". In this case every customer bought at least about 200 products. Figure 1 shows both customer and product coordinates; customers correspond to small squares, products to small $+$'s.

Figure 1



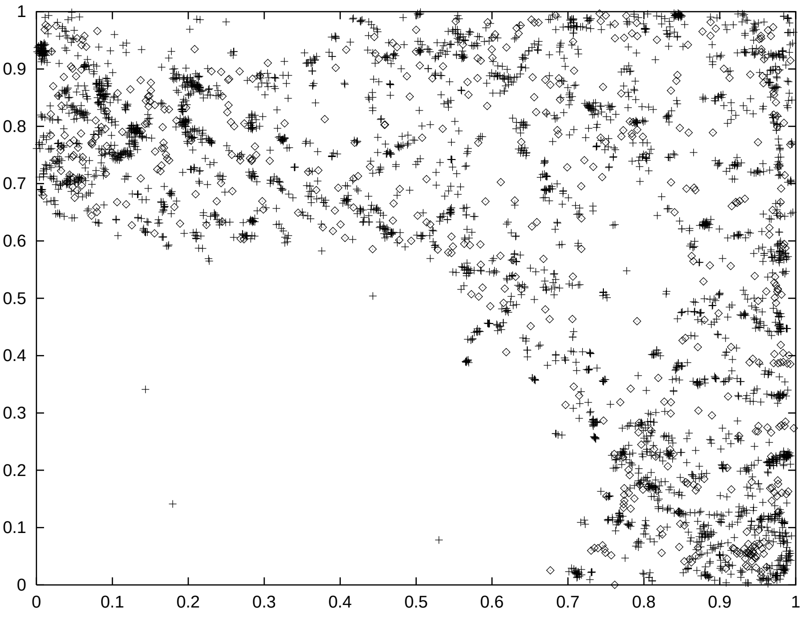
As a second example (see Figure 2) we consider a situation where we achieved a total error of 21015, with 20773 "only model bought". The total number of sales was 117997. This example shows the importance of repeated runs — the parameters were exactly the same as in the previous example, but this time the unit square was only partially used. The result is not really bad, but if one fills the whole square, as in the previous example, a better result is possible. Note that the coordinates found are situated in a rectangular shape, suggesting that the regained coordinates have almost the same positions relative to each other as the original ones. This illustrates the fact that there are some degrees of freedom (i.e. rotations and scaling) in the behaviour of the model.

Figure 2



As a final example for dimension $k = 2$ we present some results of an experiment with few sales. In this case only 14985 products were sold. The total error was 24094, with 20850 “only model bought”. Figure 3 is representative for many experiments: the final error is large, whereas customers and products are not evenly distributed in space, contrary to the artificially generated ones.

Figure 3



From the experiments we concluded that the best results were obtained when customers bought very many products. Intuitively customers that buy only a few products are not representative, and from a marketing point of view, these customers are perhaps not the most interesting ones. In the sequel we therefore restrict our attention to customers buying at least 40 products.
In the following table we give some results in the case of artificial data for higher dimensions, where every customer bought about 50 products. The number of customers considered was near to 200.

name	dimension	number of cycles	"both bought"	"only reality bought"	"only model bought"
A	4	2820	6868	493	83
B	5	2460	12323	1294	267
C	6	2400	12348	1768	331
D	7	2000	13422	1648	345
E	8	2060	12072	356	207
F	9	1480	10956	338	91
G	10	1680	6695	25	12
H	10	1760	6115	38	17
I	10	1320	24426	4259	1042

Here the experiments G, H and I had slightly different parameters. Some runs were almost perfect. We may conclude that the neural network is capable of fitting the coordinates in the case of artificial data.

6 Results on Real Data

In this section we present some results from experiments on real data. As a running example we use a real sales list consisting of $n = 1368$ customers, who bought 32523 products. The number of different products was $m = 10381$. We restricted our attention to customers buying at least 40 products. These 228 customers (17% of the total number of customers) bought $t = 11769$ products (36% of the total number of products sold). As this example will show, reasonable results can be obtained.
In the first table we present some simple theoretical values. We denote by "k-wrong" a situation where the total number of products bought matches the corresponding number from the sales list exactly, but precisely k products are replaced by incorrect ones. Of course, perfect is 0-wrong.

name	dimension	"both bought"	"only reality bought"	"only model bought"
naive	any	0	t	0
perfect	any	t	0	0
k-wrong	any	$t - k$	k	k

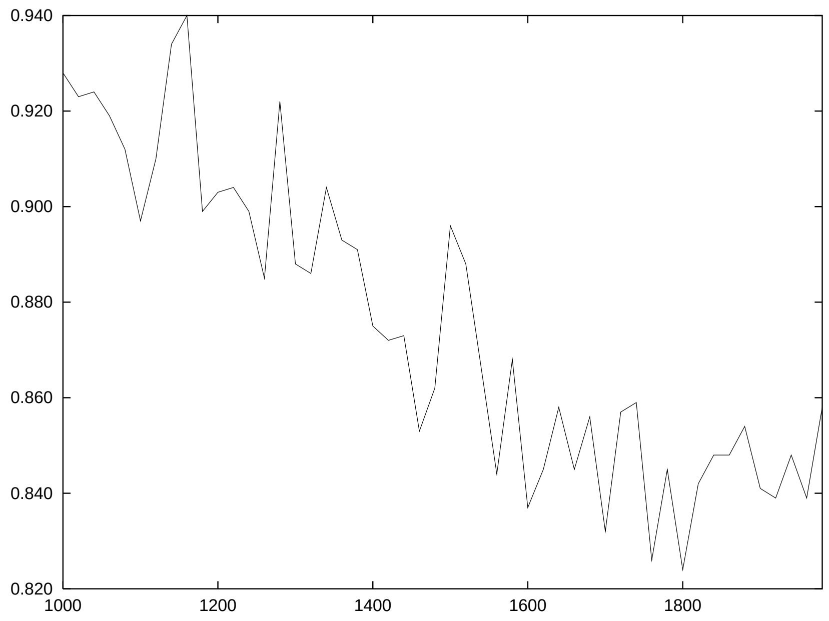
In the second table we show results from interesting runs using real data. Note that "both bought" and "only reality bought" always add to $t = 11769$. The error is a combination of the columns "only reality bought" and "only model bought".

name	dimension	number of cycles	"both bought"	"only reality bought"	"only model bought"
X	9	2340	6468	5301	1211
Y	9	1360	5512	6257	668
A	10	1820	7072	4697	1138
B	10	1440	6010	5759	499
C	10	1760	7990	3779	5700

As an illustration two experiments, X and Y, in dimension 9 have been added. The results arise from slightly different settings of parameters, and from repeated experiments with the same parameters.

Experiment A shows a relatively good result. An average customer buys 52 products and the model correctly buys 31 of these, where instead of the remaining 21 it buys 5 incorrect products. For experiment B this last number is lowered to 2, but only 26 products are properly handled on average. In experiment C we see that on average even 35 products out of 52 are correctly predicted by the model; raising "both bought" seems only possible at the cost of raising "only model bought" too.
As a visualisation of experiment A we plot the error against the number of cycles. In Figure 4 we show the "relative error": the sum of "only reality bought" and "only model bought" divided by "both bought".

Figure 4



From the experiments we noticed that the best results were achieved when using the Euclidean distance d, defined by $d(x, y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ for vectors $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ in k-space. We also experimented with Manhattan-like distances (taking the sum of the absolute values of $x_i - y_i$), but results were not very promising in that case: the errors obtained were much larger.

7 Conclusions and Further Research

In this paper we described models to understand and analyse customer choices. Using techniques from neural networks, we were able to obtain reasonable results for real data in the case of customers buying lots of products.
At the moment it is not possible to conclude whether or not these models reflect real life customer choices. In order to do so, we have to examine several topics. Of course, it would be unrealistic to expect customers to behave exactly as in a model, but we do hope to discover underlying trends. Feedback from domain experts may reveal the possible meaning of the dimensions. We mention some topics of interest:

- The theoretical power of the two models should be compared. For instance, what is the complexity of the decision problems involved?
- The models can be adjusted in several ways. As an example, it is possible to account for situations where customers can buy more than one item per product.

- Our program —as usual for neural networks— has many parameters. We have to do lots of experiments to obtain a fine-tuned system. We also have to give a theoretical basis for several choices involved here.
- At the moment a run takes several hours on a workstation, due to the number of iterations involved. The algorithm has to be optimised in order to get acceptable response times.
- The method has to be applied to real life data from different branches. Perhaps this has some influence on the behaviour. We would also like to examine the behaviour on totally random data.
- Cluster analysis has to be applied to the product coordinates supplied by good runs. Do these coordinates, or these clusters, have any significance in real life? In general, we have to apply statistical methods to clarify the results.
- We would like to examine how well the coordinates provided by the system correspond with the real coordinates in case of artificial data.

From a marketing point of view we also have some remarks:

- We see that a certain relationship between products can be analysed, which will have an impact on the organisation of the warehouses, so that the picking process of different products can be set up more efficiently.
- When finding a new clustering of products it will also be possible to describe these clusters and to analyse what the functions of these products are to the customers who ordered them. In doing so the company will get a deeper insight in the similarities and differences of their customers.
- It is expected that given a time series of analyses, differences in seasons can also be analysed. By doing so the company will be able to adapt better to seasonal and other fluctuations than it does today.

All in all we think we have some promising methods that still require further research.

References

[1] U. Fayyad and R. Uthurusamy (editors), Data Mining and Knowledge Discovery in Databases, Communications of the ACM, Volume 39, No. 11, Special issue, November 1996

[2] M.R. Garey and D.S. Johnson, Computers and Intractability, Freeman, 1979

[3] S. Haykin, Neural Networks: a Comprehensive Foundation, MacMillan, 1994

[4] J. Hertz, A. Krogh and R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, 1991

[5] M.C. van Wezel, J.N. Kok and K. Sere, Determining the Number of Dimensions Underlying Customer-Choices with a Competitive Neural Network, Proceedings of the IEEE International Conference on Neural Networks (ICNN'96), Volume 1, 484–490, 1996