Working Paper

The Applicability of Neural Nets for Decision Support

E.H.L. Aarts
J. Wessels
P.J. Zwietering

WP-93-005
January 1993

The Applicability of Neural Nets for Decision Support

E.H.L. Aarts
J. Wessels
P.J. Zwietering

WP-93-005
January 1993

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

There are different paradigms for decision support. The most usual one has, as core-activity, the modelling of the basic processes together with the possible decisions and their effects. In this approach, the goal of the modelling is to find a problem formulation which truly represents reality and allows evaluation with the available methodology. Although the modelling paradigm can point at many successes, it does not satisfy in all cases. For instance, it is not always possible to find a workable compromise between "truly representing reality" and "analyzability of the resulting models". There are also cases where the basic processes are so badly understood that proper modelling is doomed to fail. One typically encounters these phenomena in some economic and in some environmental processes. In such cases, the black-box paradigm might be useful. In this paradigm, one relaxes the ambition to model the basic processes. According to the black-box paradigm, one takes some parametrized black-box and tries to fit the parameters in such a way that the black-box shows a sensible behavior in at least a representative set of cases. It appears that neural nets are good candidates for being used as black-boxes in decision situations, because of their learning capabilities and their apparent qualities in pattern recognition.

The present paper investigates how these capabilities may be exploited for decision support. The results show that the approach is very promising.

The Applicability of Neural Nets for Decision Support

E.H.L. Aarts[1,3] J. Wessels[1,2] P.J. Zwietering[1]

Abstract

The aim of this paper is to discuss the possible role of neural nets for decision support. The discussion will be conducted along two connected lines. The first line regards the possibilities to solve combinatorial optimization problems with multilayered perceptrons. Particular attention is paid to the required complexity of such neural nets. The second line regards the use of neural nets as a kind of black-box decision mechanism for decision problems for which the underlying processes are difficult or even not well understood. For direct use of the results of the first line it would be essential to have cheap neuro-based computing aids available. However, even without the availability of such aids, the results are useful for supporting the second line. The second line itself has its value independent of the availability of special neural computing aids. Its essence is that it provides an alternative paradigm for decision support, which can also be simulated on traditional computing equipment.

1 Introduction

In recent years, neural nets have become very popular. The main reasons for this rise in popularity are, firstly, the apparent successes of neural nets for the solution of problems with a pattern-recognition-related aspect and, secondly, the possibility to tune neural nets by learning from examples.

In this paper it will be investigated how and when neural nets can play a role as a tool for decision support. With the term "decision support" we will indicate all help that can be given for decision making with respect to the management of systems, the design of control systems or the planning of activities. So, we don't include technical applications like automatic reading of cheques in a bank.

The essential feature of the neural net approach to decision making is that it is a black-box approach, which means that one does not try to model the underlying processes, but only looks for a tuning of the parameters of the neural net such that the black-box mimics sensible behavior. One may expect that such an approach can lead to useful results when, firstly, the task of the black-box is essentially pattern recognition and, secondly, the decision problems are too complicated for the traditional model building approach and, moreover, it is relatively easy to say what would have been the right decision a posteriori. The latter feature makes it easy to construct learning pairs for the tuning of the neural nets.

[1]Technical University Eindhoven, Eindhoven, The Netherlands.
[2]International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria.
[3]Philips Research Laboratories Eindhoven, Eindhoven, The Netherlands.

We will illustrate the possibilities of using neural nets for decision support by treating a simple type of production planning problem. Many production planning problems can be formulated as combinatorial optimization problems. However, most production planning problems also have a dynamic aspect, which is not taken into account in the combinatorial optimization formulation. Usually the dynamic aspect implies that at the beginning of the planning there is a lack of information and, in the course of the process, more information becomes available gradually. In production planning problems, the external demand often represents such a dynamic aspect, but also available capacities or required resources may be uncertain in the beginning. Modeling of such features is, in several cases, extremely difficult, particularly, if the underlying processes are not well-understood. The use of such models for optimization is also complicated and, practically, only feasible for highly simplified models. Afterwards, it is often quite easy to conclude what should have been done. Therefore, it seems worthwhile to investigate whether under these circumstances neural nets would be able to use their pattern-recognition-abilities in such a way that they discern the right decision at the right time.

We will restrict attention to a relatively simple type of neural nets, namely multi-layered perceptrons with hard-limiting response functions. First, we will investigate the capabilities of multi-layered perceptrons for solving combinatorial optimization problems. Afterwards, we will demonstrate how the dynamic aspect might be included. The paper is organized in the following way. Section 2 contains a definition of multi-layered perceptrons and some results regarding their classification capabilities, since classification is what multi-layered perceptrons with hard-limiting response functions essentially do. In Section 3, combinatorial optimization problems are introduced and related to classification problems. Section 4 summarizes some results regarding the capabilities of multi-layered perceptrons for solving combinatorial optimization problems. In Section 5, a dynamic aspect is added to the well-known production planning problem of Wagner and Whitin. This is done by removing the finite time horizon and introducing unknown demands for time periods farther away. These demands become known gradually as time goes by. Section 6 contains some conclusions and remarks.

2 Multi-layered Perceptrons

Multi-layered perceptrons consist of a number of nodes arranged in some layers (cf. figure 1 for an example with 3 layers). The nodes are simple processors, which get information from "below", i.e. the first layer gets information from the outside world (the x_i in figure 1) and the nodes in layer n get information from the nodes in layer $n-1$ ($n \geq 2$). So, all nodes proceed the result of their processing step to nodes in the next higher layer, i.e. all links emerging from one node transport the same information at the same time. The nodes in the top-layer communicate their results to the outside world (the y_j in figure 1). The nodes process the incoming information as follows: the values on incoming links are multiplied by a link-specific weight and added, the outgoing links all get the same value, determined by a function on the sum of weighted incoming values. In the case of the hard-limiting response function for a node with incoming values $u_i, i = 1, \cdots, n$, and outgoing value v, this would imply:

$$\text{if } \sum_{i=1}^{n} a_i u_i \geq b \text{ then } v = 1$$

$$\text{if } \sum_{i=1}^{n} a_i u_i < b \text{ then } v = 0.$$

Here $a_i, i = 1, \cdots, n$, are the weights and b is a threshold value.

We will only consider hard-limiting response functions and hence the inputs x_i may be real valued, but the outputs y_j are necessarily zero or one. One reason for this choice is that it fits well with use for planning purposes; a second reason is that one may expect that this choice facilitates the analysis; a third reason is that multi-layered perceptrons learn relatively fast.

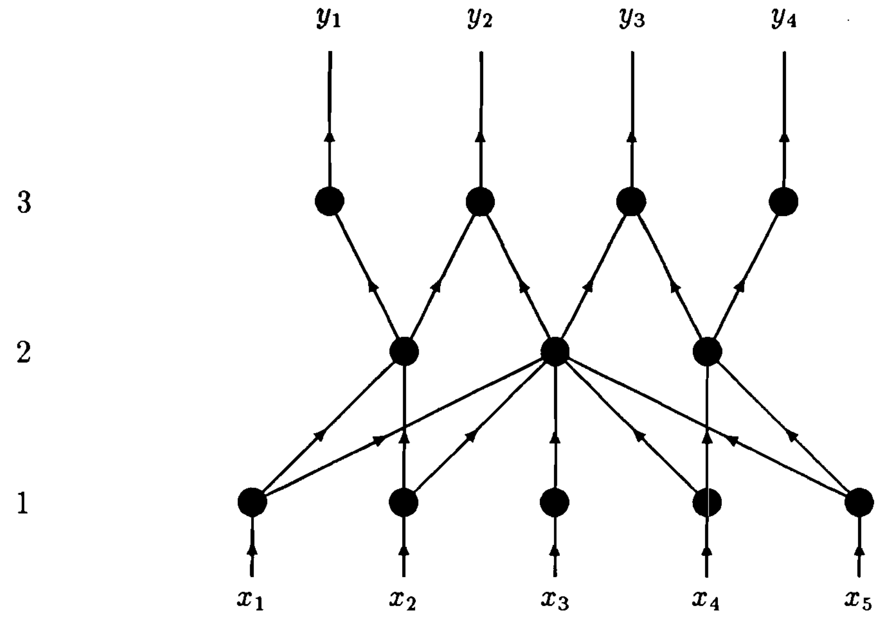

Figure 1: *A multi-layered perceptron with 3 layers of nodes; the x_i are the inputs of the perceptron and the y_j the outputs.*

Let us consider an m-layered perceptron with n inputs $x_i, i = 1, \cdots, n$, and k outputs $y_j, j = 1, \cdots, k$ and with hard-limiting response functions on the nodes. Then the set of input vectors $\Re^n$ consists of vectors $(x_1, \cdots, x_n)$ which result in $y_1 = 1$ and of vectors $(x_1, \cdots, x_n)$ which result in $y_1 = 0$. In other words, $\Re^n$ is partitioned in 2 sets A_1 and $A_1^* = \Re^n \backslash A_1$ such that

$$y_1 = 1 \quad \text{for} \quad (x_1, \cdots, x_n) \in A_1$$

$$y_1 = 0 \quad \text{for} \quad (x_1, \cdots, x_n) \in A_1^*$$

Similarly, $y_j, j = 2, \cdots, k$, signalises whether $(x_1, \cdots, x_n) \in A_j \subset \Re^n$ or not.

Summarizing, we may conclude that the effect of putting a vector $(x_1, \cdots, x_n)$ into this m-layered perceptron is, that the vector $(y_1, \cdots, y_k)$ indicates to which of the sets $A_j, j = 1, \cdots, k$, the input vector belongs. In this sense the perceptron performs a classification task: it classifies any possible input vector with respect to the sets $A_1, \cdots, A_k$.

Actually, the consequence of the preceding reasoning is that, if we want to use a perceptron for some task, then this task is (explicitly or implicitly) translated into some classification problem. So, it is important to know which classification problems can be solved by multi-layered perceptrons.

Several authors show that a given subset in $\Re^n$ can be classified by some two-layered perceptron with arbitrary precision (cf. Cybenko [3], [4], Funahashi [7], Hornik et al. [8]). However, approximate classification is difficult to translate to the original problems, therefore it is worthwhile to obtain more knowledge about the exact classification capabilities of multi-layered perceptrons.

Consider therefore m-layered perceptrons with n inputs and one output. Such perceptrons exactly classify one subset of $\Re^n$.

Denote by C_m the class of subsets of $\Re^n$ which can be classified exactly by some m-layered perceptron. Formally, we call a set $A \subset \Re^n$ classified by an m-layered perceptron if for some m-layered perceptron we have

$$\begin{aligned} &\text{if } (x_1, \cdots, x_n) \in A, \text{ then } y = 1 \\ &\text{if } (x_1, \cdots, x_n) \notin A, \text{ then } y = 0 \end{aligned}$$

Then we have the following results:

Lemma 1 (cf. Minsky and Papert [12], Lippmann [11], Zwietering et al. [22])*:*
Let $\{V_i | i = 1, \cdots, l\}$ be a collection of subsets with

$$V_i \in C_m \text{ or } V_i^* \in C_m \text{ for all } i,$$
$$\text{then } \bigcap_{i=1}^{l} V_i \in C_{m+1}$$

Theorem 1 (cf. Zwietering et al. [22])*:*

$$\begin{aligned} C_m &\subset U \text{ for } m \geq 1 \\ C_m &\supset U \text{ for } m \geq 3, \end{aligned}$$

where U is the collection of subsets of $\Re^n$ which can be written as finite union of pseudo-polyhedra; a pseudo-polyheder is the intersection of a finite number of open or closed linear half-spaces.

Corollary 1

$$C_m = U \text{ for } m \geq 3.$$

From these results one may conclude that the capabilities of 4-layered perceptrons do not exceed the capabilities of 3-layered perceptrons, although it is still possible that a 4-layered perceptron performs the same task as a 3-layered perceptron but needs less nodes for it.

From Lemma 1 we may conclude that pseudo-polyhedra themselves may be classified by 2-layered perceptrons. So,

$$P \subset C_2 \subset U$$

where P denotes the set of pseudo-polyhedra in $\Re^n$ (cf. Lippmann [11]). Indeed, C_2 is only a subset of U, since there is a necessary condition for classifiability by a 2-layered perceptron which excludes a substantial subset of U.

Theorem 2 (cf. Zwietering et al. [23]):
$V \subset \Re^n$ can only be classified by a 2-layered perceptron if there do not exist a set $W \in C_1$ and open balls B_1 and B_2 such that

$$\begin{array}{lll} \phi & \neq B_1 \cap W^0 & \subset V \\ \phi & \neq B_2 \cap W^0 & \subset V^* \end{array} \quad \text{and} \quad \begin{array}{lll} \phi & \neq B_1 \cap W^* & \subset V^* \\ \phi & \neq B_2 \cap W^* & \subset V \end{array}$$

Here W^0 indicates the interior of W.

Figure 2 indicates 3 sets which cannot be classified with a 2-layered perceptron according to Theorem 2. For practical purposes it is useful to check whether one of these structures is present or not. The next theorem gives a sufficient condition for sets in $\Re^n$ to be classifiable by a 2-layered perceptron. This theorem can be used to prove that C_2 is essentially larger than P. A nice feature of this theorem is that it can be verified algorithmically.

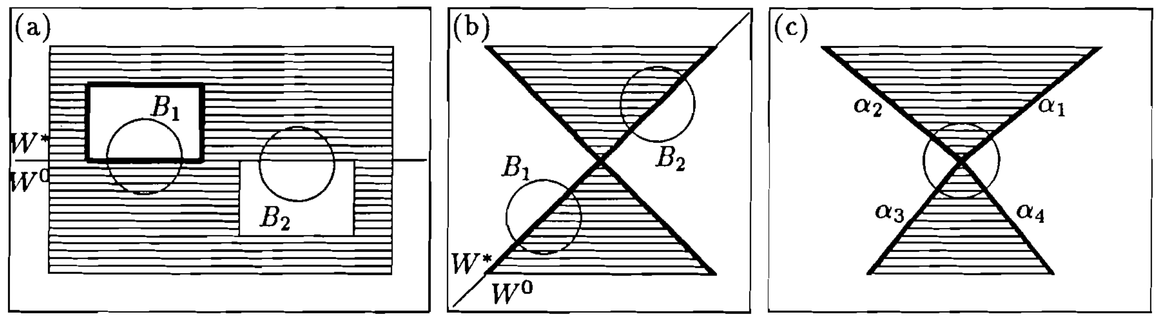

Figure 2: *Three subsets that cannot be classified exactly by a 2-LP. Note that solid boundary lines do and thin boundary lines do not belong to the presented sets.*

Theorem 3 (cf. Zwietering et al. [23]):

$$\begin{array}{rl} \textit{If} & V_i \in P,\ i = 1, \cdots, l, \\ \textit{then} & V_1 \backslash (V_2 \backslash \cdots (V_{l-1} \backslash V_l) \cdots) \in C_2. \end{array}$$

For an algorithm which verifies if a given subset of $\Re^n$ can be written in the form shown in theorem 3, we again refer to [23].

In fact, the proofs of the theorems as stated here also give indications of required numbers of nodes, but we will not go into that detail here.

3 Combinatorial Optimization

In many planning problems there is an essential combinatorial aspect. Therefore, it is useful to obtain insights in the capabilities of neural nets with respect to solving combinatorial optimization problems. In this paper we will try to obtain the required insight for multi-layered perceptrons with hard-limiting response functions.

From the preceding section it is clear that the essential step will be to translate combinatorial optimization problems into classification problems.

What do we mean by solving a combinatorial optimization problem by a neural net? We hope to find a multi-layered perceptron for a particular type of combinatorial optimization problem such that all instances of that problem are solved by that multi-layered perceptron. In this terminology, two instances of one problem only differ with respect to input data. For example, the travelling salesman problem has as input data a square matrix representing the distances between the cities involved. So, each square matrix with non-negative entries represents an instance of the travelling salesman problem. Hence, what we are looking for is a multi-layered perceptron which can be fed with an arbitrary square distance matrix and which gives the optimal route for the related instance of the travelling salesman problem.

From the above problem statement, we can already derive some constraints. In the first place, we have to accept that multi-layered perceptrons have a fixed input dimensionality. As a consequence, we better restrict attention to problems with a fixed size for the input data, i.e. "the travelling salesman problem for m cities" rather than " the travelling salesman problem". In the second place, the problem should be formulated in such a way that the output of the perceptron, consisting of zeros and ones, determines the solution. As a consequence, we need problem formulations in which the decisions are represented by zeros and ones.

Formally, we can now proceed as follows (cf. [22]):

Definition: 1

A (fixed sized) combinatorial optimization problem is a triplet (I, F, C), where I is a subset of $\Re^n$ for some n, each $x \in I$ will be called an instance of the problem and represents the input data, F determines for every $x \in I$ a subset $F(x)$ of $\{0,1\}^k$ for some fixed k of feasible solutions for the instance x, and $c(\cdot\,; x)$ maps for fixed x the set $F(x)$ in $\Re$ and represents the criterion function which should be minimized for fixed x.

For convenience it is supposed that c is chosen in such a way that for each x the minimizing solution is unique. This assumption is not restricting generality, since any preference structure can always be made strict in a simple way.

Theorem 4 (cf. [22])*:*

The combinatorial optimization problem (I, F, c) is solved by solving the classification problem in $\Re^n$ for the sets $(A_1, \cdots, A_k)$, where

$$A_j = \{x \in I | (\arg \min_{y \in F(x)} c(y; x))_j = 1\},$$

i.e. instance $x \in I$ is solved by the classification result of vector x.

The next question to be solved is about the nature of the classification problem of theorem 4: if $A_j \in C_m, j = 1, \cdots, k$, then the classification problem as well as the underlying combinatorial optimization problem can be solved by an m-layered perceptron. The following theorem indicates that, indeed, most combinatorial optimization problems can be solved by 3-layered perceptrons.

Theorem 5 (cf. [22])*:*
Let (I, F, c) be a combinatorial optimization problem with $I = \Re^n$.
This problem can be solved by a 3-layered perceptron with n inputs and k outputs if

$$\{x \in I | c(y;x) \le c(z;x)\} \in C_1 \textit{ for all } y, z$$

$$\{x \in I | y \in F(x)\} \textit{ is a polyhedron for all } y$$

In this way it is shown that piece-wise linearity of both the cost function and the feasibility constraints is sufficient for exactly solving the problem with a 3-layered perceptron. It is never necessary to use a 4-layered perceptron, however, it appears that in some cases a solution via a 4-layered perceptron requires much less hidden units. This is particularly the case if the sets $F(x)$ really depend on x. This question of the required number of hidden units is discussed in some more detail in [22]. The general conclusion, however, may be that the construction of theorem 5 will require large numbers of hidden units, since it is essentially based on enumeration. Therefore, it remains an interesting problem to find more efficient perceptrons for particular combinatorial optimization problems.

4 Sorting and Lot-sizing

In the present section some results will be summarized regarding two simple types of combinatorial optimization problems, namely sorting and lot-sizing. Let us start with sorting.

In its standard formulation the sorting of n real numbers $x_1, x_2, \cdots, x_n$ amounts to finding a permutation π, where $\pi(i)$ indicates the position of x_i, such that

$$x_{\pi(i)} \le x_{\pi(i+1)} \text{ for } i = 1, 2, \cdots, n-1.$$

This formulation can easily be transformed into the formulation of a combinatorial optimization problem by representing the permutation by an $n \times n$-matrix with $(0-1)$ -entrances and introducing a minimization criterion. The matrix-entrance y_{ij} is defined as 1 if $\pi(i) = j$ and 0 otherwise. As criterion we may choose

$$c(y;x) = -\sum_{i,j=1}^{n} v_i x_j y_{ij}$$

with $v_i, i = 1, \cdots, n$ an arbitrary strictly increasing sequence.

The solution of this combinatorial optimization problem can be made unique by requiring that in case of equal entries the ones with lower indices come first, but we will not work this detail out.

It can easily be shown that the sorting problem satisfies the requirements of theorem 5 and therefore:

Theorem 6 (cf. Zwietering et al. [24])*: The sorting problem for n real numbers can be solved by a 3-layered perceptron.*

However, the perceptron obtained in this enumerative way is quite large. It contains $O((n!)^2)$ nodes.

By considering the constructed perceptron more carefully it appears that it does more than necessary and, hence, it can be reduced in size. By first reducing the number of comparisons performed by the first hidden layer to $\frac{1}{2}n(n-1)$ and subsequently reducing the number of nodes in the second layer we obtain:

Theorem 7 (cf. Zwietering et al. [25]):
The sorting problem for n real numbers can be solved by a 3-layered perceptron with $\frac{1}{2}n(n-1)$ nodes in the first hidden layer and n^2 nodes in the second hidden layer as well as in the output layer. In total this perceptron needs $\frac{1}{2}n(5n-1)$ nodes.

This 3-layered perceptron has a strong affinity to the 5-layered perceptron proposed for the same problem by Chen and Hsieh [2]. It is also strongly related to Preparata's parallel sorting algorithm [15]. This is the more remarkable since the perceptron of theorem 7 is the result of applying reduction techniques on a perceptron obtained by a standard construction.

One might wonder whether it is possible or not to find a 2-layered perceptron which does the same job. However, theorem 2 can help in proving that two layers are not sufficient (cf. [24]). Actually, it can be shown that it would be necessary to classify a set similar to the one in Figure 2(b) by a 2-layered perceptron which is impossible according to theorem 2.

It can also be proved that the number of nodes in the first hidden layer cannot be diminished. So, the required number of hidden nodes in a 3-layered perceptron solving the sorting problem for n real numbers is $O(n^2)$ (cf. [25]).

It is also possible to formulate the sorting problem in a less ambitious way. In such a formulation it may very well be possible to solve the problem by a 2-layered perceptron (cf. [24]).

The other problem to be treated in this section is a simple version of the economic lot-sizing problem. In its original form the problem was posed by Wagner and Whitin [20] and transformed into a shortest path problem, which could be solved by dynamic programming. We use the following formulation:

> For n periods it has been specified how much of a product has to be made available: d_i for period $i, i = 1, \cdots, n$. The marginal production costs are p per unit of product, but there is also a fixed set-up cost r for each period with actual production. On the other hand it is possible to avoid set-ups by combining the production of demands for different periods as long as the required amounts are available in time. Each unit of product is charged with holding costs h for any period it is produced earlier than necessary.

Wagner and Whitin's dynamic programming algorithm solves this problem in $O(n^2)$ time (cf. [20] and Evans [5] for an efficient implementation). It is even possible to solve this problem in $O(n \log n)$ time by exploiting its special features (cf. Wagelmans et al. [19]). By adding capacity constraints, the problem would become much harder from a computational point of view (cf. Bitran, Yanasse [1], Florian et al. [6], Salomon [17]).

It will be clear that the lot-sizing problem, as formulated here, has a cheapest solution of the following form: if there is production in a period, then the lot-size will be the total demand for a number of subsequent periods; moreover, there will only be production if stock is empty. Note that the marginal production costs p are irrelevant for the minimization. As a consequence, it suffices to determine in which periods there will be production since the lot-sizes follow from these decisions. In this way it is easy to give a $0-1$ formulation of the problem:

$$\begin{array}{rcl} x \in I & = & [0,\infty)^{n+1} \text{ with } x_i = d_i \text{ for } i = 1,\cdots,n \text{ and } x_{n+1} = r/h \\ y \in F & = & \{0,1\}^n \text{ with } \begin{array}{rcl} y_i & = & 0 \text{ if there is production in period } i \text{ and} \\ y_i & = & 1 \text{ if there is no production in period } i. \end{array} \\ c(y;x) & = & \sum_{i=1}^{n}\sum_{j=1}^{i}(\prod_{l=j}^{i} y_l)x_i - x_{n+1}\sum_{i=1}^{n} y_i \end{array}$$

According to theorem 5, this problem can be solved exactly by some 3-layered perceptron. A neural net which does the job can be found straightforwardly by applying the construction method of [22]. Unfortunately, however, this 3-layered perceptron has a number of hidden units which is exponential in n. One might hope that in this case there exists also an essentially more efficient 3-layered perceptron since the problem is in POLYLOGSPACE (cf. Kindervater [9]) and it has been shown in [22] that a perceptron with a polynomial number of hidden units may only be expected under that condition.

It is definitely not possible to solve the problem for $n \geq 3$ with a 2-layered perceptron since, we could have to classify a set like the one in figure 2(c), which is impossible with two layers according to theorem 2 (in [26] it is shown for an example with $n = 3$ that such a classification problem does actually occur).

5 Lot-sizing with Realistic Dynamics

Although the lot-sizing problem of section 4 is usually called "dynamic", its dynamics are rather poor and only consist of the variability of demand over time for a finite number of periods, but with given values for the demands. For practical applications this implies that the world is supposed to stop after the few periods for which the demand is known.

We need an approach in which it is accepted that new information will come in after the first decision has been implemented, so that decisions are taken under uncertainty.

The simplest modeling approach to this more realistic situation would be to suppose that from period $n+1$ onwards the demands are independent realizations of one random variable with a given distribution. One might suppose that realization d_i for $i > n$ becomes known at the beginning of period $i-n$. In this way the demand would always be known for a time-horizon of n periods. This problem, with the discounted or average cost criterion, could be formulated as a Markov decision problem and solved as such. However, the state space of the problem would be rather large. A simple approximation could be obtained by replacing the random demands by their constant expected value (cf. Van Nunen, Wessels [13], where this idea is used for a decomposition approach for the case of several products and a capacitated production unit).

A disadvantage of this type of approach is that it only works for very simple models of the demand generating process, whereas, in reality, the demand generating process is frequently rather complicated and not well-understood. Therefore, it seems sensible to explore the possibilities of a black-box approach based on multi-layered perceptrons, exploiting their proven capabilities for solving combinatorial optimization problems, for predicting real-world time series (cf. Weigend et al. [21]), for recognizing patterns (cf. Pao [14]) and their conjectured capabilities for adapting to changing circumstances. The latter type of capabilities is strongly related to the learning capabilities (cf. Rumelhart et al. [16]), which can be extended to relearning.

In the rest of this section, we will describe some experiments with demand generators of increasing complexity. The experiments are run through 1000 periods, but the decisions have to be taken with demand information for 5 periods. Only known demand can be produced. In the experiments 3 methods are compared:

a) a rolling horizon version of Wagner and Whithin's method;

b) a rolling horizon version of Silver and Meal's heuristic;

c) a method using a multi-layered perceptron.

ad a) In the rolling horizon version of Wagner and Whitin's method, one computes the optimal solution for the next 5 periods as if the world was to stop. This is repeated every period which starts with positive demand and zero-stock. Only the first decision of the optimal solution is implemented. If the usual lot-size is essentially less than the demand for 5 periods, one may expect that the future has only relative importance for the decisions.

ad b) Silver and Meal [18] have published a simple heuristic for the problem with a finite planning horizon which can easily be transformed into a rolling horizon version. The idea is the following:

$$\text{Compute subsequently } a_i = \frac{1}{i}(r + h\sum_{l=0}^{i-1} l d_{t+l}) \text{ for } i = 1, \cdots, 5$$

and choose the value of i for which $a_{i+1} \geq a_i$ for the first time. If this does not occur choose $i = 5$ at time t if there is no stock available and $d_t > 0$.

In [26] it is shown that this procedure can be represented by a 1-layered perceptron.

ad c) A 2-layered perceptron is used. One of the reasons to use two layers is that the popular Silver-Meal heuristic is essentially a 1-layer approach. So the question is whether more layers really help or not. Another reason is that 2-layered perceptrons are supposed to learn faster than 3-layered perceptrons. The 2-layered perceptron is adjusted by presenting a number of learning pairs in different rounds using back propagation in a more or less standard way (cf. [26] for details). The perceptron is not adapted during the experiment. We use 10 hidden units although the difference with 5 hidden units is not essential. The learning is based on optimal solutions for 10 periods in order to account for learning to anticipate on the future.

The results of the three methods are compared with the optimal solution for the case that the demands for all 1000 periods would have been known in advance. This solution is computed via dynamic programming. For the three methods the average loss in costs over 1000 periods is indicated for a series of experiments with the same model.

For more details of the experiments compare Zwietering et al. [26] and Van Kraaij [10]. In all experiments reported here the value of h is 1 and the value of $r = 100$.

A. In the first experiment the demands are supposed to be independent and identically distributed according to a uniform probability density on [5,90].

In this case the dynamic programming solution for given demands over 1000 periods gives the following relative frequencies for the lot-sizes in the optimal solution.

1	period	:	0.15
2	periods	:	0.56
3	periods	:	0.24
4	periods	:	0.05
			1.00

So, one may expect that the rolling horizon version of Wagner and Whitin's method works fine. Indeed, it only looses 0.55% on the total costs in several experiments.

The rolling horizon version of the Silver and Meal heuristic looses 2.54%, which is more than expected, given the fame of the heuristic.

The performance of the 2-layered perceptron is much better, but not as good as the rolling horizon Wagner and Whitin. To give an indication, the results for different numbers of learning pairs, different numbers of hidden units, and different lengths of time on which the learning solutions are based, have been displayed in table 1.

In this experiment the demand process is very simple, nevertheless the 2-layered perceptrons perform already quite well. Of course one should consider that both other approaches are much simpler.

number of learning pairs:		500	800	1200
time length 5	5 hidden units	1.48	1.36	1.14
	10 hidden units	1.42	1.26	1.10
time length 10	5 hidden units	0.97	0.89	0.88
	10 hidden units	0.94	0.91	0.91

Table 1: *For experiment A the losses in costs with respect to the ideal solution are indicated in percentages for different set-ups with a 2-layered perceptron.*

B. In order to introduce some complexity in the demand pattern, we introduce a cyclic demand process by taking demand alternatively from uniform distributions on [5, 47.5] and [47.5, 90] with a random choice for the starting distribution.

In this case the ideal solution is always purely cyclic with cycle length 2.

All three approaches find the ideal solution exactly.

C. The next step is to keep demand cyclic, but with overlapping ranges: [5, 70] and [25, 90]. The average losses now become:

Wagner/Whitin	:	0.50%
Silver/Meal	:	1.57%
2-layered perceptron	:	0.80%

D. For a cyclic demand pattern with cycle length 3 on the ranges [5, 40], [25, 65], [50, 90] the average losses become:

Wagner/Whitin	:	0.93%
Silver/Meal	:	5.50%
2-layered perceptron	:	0.50%

Apparently, it becomes more difficult for the other approaches to cope with the complex pattern, but the perceptron feels fine.

E. For a cyclic demand pattern with cycle length 6 on the ranges [5, 40], [5, 40], [25, 65], [25, 65], [50, 90], [50, 90] we get the following average losses compared to the ideal solution:

Wagner/Whitin	:	4.71%
Silver/Meal	:	2.75%
2-layered perceptron	:	1.00%

In this case, the Wagner and Whitin approach works worse because larger production lots become more reasonable. An extra experiment with the Wagner/Whitin version for 10 periods in which the 5 extra periods get an estimated demand according to the modeled pattern, leads to a diminished loss of 0.64%. Of course, this experiment represents an unrealistic situation, but it shows the relevance of some extra information on the demand pattern. It also shows that the 2-layered perceptron does the forecasting very well.

F. The last experiment reported here is based on a demand generation with a still more complex pattern.

There are supposed to be three ranges for the demand, namely [5, 40], [25, 65], [50, 90]. A Markov chain determines which range is used: the first, the second or the third. Transition probabilities are $P_{12} = P_{31} = 1$, $P_{21} = 1 - P_{23} = 0.2$.

The experiments show the following average losses with respect to the ideal solution:

Wagner/Whitin	:	4.49%
Silver/Meal	:	1.27%
2-layered perceptron	:	0.90%

6 Conclusions and Remarks

In section 5 we saw indications that, indeed, multi-layered perceptrons might be used well for decision support in a situation with complex and, perhaps, badly understood background processes. A strong point of the perceptron approach is that the learning can be based on optimal posterior solutions which are computed after realization of the background processes. In the examples, the computation of posterior solutions is simpler and, particularly, requires less modeling assumptions.

A weak point of the perceptron approach is the necessity of having large numbers of learning pairs. It seems that current research can help in speeding up the learning process, but, most probably, this point will remain an essential weakness in any method. This weakness becomes particularly apparent if one wants to extend the learning to relearning for the incorporation of adaptivity. Nevertheless, the main conclusion may be that the black-box approach has promising features for use in decision support for at least a particular type of situations.

References

[1] G.R. Bitran and H.H. Yanasse: Computational complexity of the capacitated lot size problem. Management Science 28 (1982) pp. 1174-1186.

[2] W.T. Chen and K.R. Hsieh: A neural sorting network with $O(1)$ time complexity. Proceedings 1990 IEEE INNS International Joint Conference Neural Networks 1 (1990) pp. 87-95.

[3] G. Cybenko: Approximation by superpositions of a sigmoidal function, Technical Report, No. 856 University fo Illinois (1989).

[4] G. Cybenko: Complexity theory of neural networks and classification problems, Proceedings of the 1990 EURASIP Workshop on Neural Networks, Sisimbra, Portugal (1990) pp. 26-44.

[5] J.R. Evans: An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. Journal of Operations Management 5 (1985) pp. 229-235.

[6] M. Florian, J.K. Lenstra, and A.H.G. Rinnooy Kan: Deterministic production planning: algorithms and complexity. Management Science 26 (1980) pp. 669-679.

[7] K. Funahashi: On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (1989) pp. 183-192.

[8] K. Hornik, M. Stinchcombe and H. White: Multilayer feed forward networks are universal approximators, Neural Networks 2 (1989) pp. 359-366.

[9] G.A.P. Kindervater: Exercises in parallel combinatorial computing, Ph.D. Thesis, Rotterdam (1989).

[10] M.J.A.L. van Kraaij: The use of neural nets for lotsizing. Master Thesis (in Dutch), Technical University Eindhoven (1991).

[11] R.P. Lippmann: An introduction to computing with neural nets. IEEE ASSP Magazine 4 (1987) pp. 4-22.

[12] M. Minsky and S. Papert: Perceptrons: An Introduction to Computational Geometry, MIT Press (1969).

[13] J.A.E.E. van Nunen and J. Wessels: Multi-item lot size determination and scheduling under capacity constraints. European Journal of Operational Research 2 (1978) pp. 36-41.

[14] Y.H. Pao: Adaptive pattern recognition and neural networks. Addison-Wesley 1989.

[15] F.P. Preparata: New parallel sorting schemes. IEEE Trans. Computation C-27 (1978) pp. 667-673.

[16] D.E. Rumelhart, G.E. Hinton, and R.J. Williams: Learning internal representations by error propagation. D.E. Rumelhart and J.L. McClelland (eds.) in: Parallel distributed processing: explorations in the microstructure of cognition. Vol. 1, Foundations MIT-Press (1986) pp. 318-362.

[17] M. Salomon: Deterministic lotsizing models for production planning. Ph.D. Thesis, Erasmus University, Rotterdam (1990).

[18] E.A. Silver and J.C. Meal: A heuristic for selecting lot size quantities for the case of deterministic time-varying demand rate and discrete opportunities for replenishment. Production and Inventory Management 14 (1973) pp. 64-74.

[19] A. Wagelmans, S. van Hoesel, and A. Kolen: Economic lot-sizing: an $O(n \log n)$ -algorithm that runs in linear time in the Wagner-Whitin case. CORE Discussions Paper No. 8922, CORE, Louvain-la-Neuve (1989).

[20] H.M. Wagner and T.M. Whitin: Dynamic version of the economic lot size model. Management Science 5 (1958) pp. 89-96.

[21] A.S. Weigend, B.A. Huberman and D.E. Rumelhart: Predicting the future: a connectionist approach. International Journal of Neural Systems 1 (1990) pp. 193-200.

[22] P.J. Zwietering, E.H.L. Aarts, and J. Wessels: The design and complexity of exact multi-layered perceptrons. International Journal of Neural Systems 2 (1991) pp. 185-199.

[23] P.J. Zwietering, E.H.L. Aarts, and J. Wessels: The classification capabilities of exact two-layered perceptrons. International Journal of Neural Systems (to appear).

[24] P.J. Zwietering, E.H.L. Aarts, and J. Wessels: The minimal number of layers of a perceptron that sorts. Journal of Parallel and Distributed Processing (to appear).

[25] P.J. Zwietering, E.H.L. Aarts, and J. Wessels: The construction of minimal multi-layered perceptrons: a case study for sorting. Neurocomputing (to appear).

[26] P.J. Zwietering, M.J.A.L. van Kraaij, E.H.L. Aarts, and J. Wessels: Neural networks and production planning. Proceedings of the fourth International Conference on Neual Networks and their Applications, Neuro-Nimes 1991 pp. 529-542.