The minimal number of layers of a perceptron that sorts

Citation for published version (APA):
Memorandum COSOR 92-06
The minimal number of layers of a perceptron that sorts

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

Eindhoven, April 1992
The Netherlands
The minimal number of layers of a perceptron that sorts

P.J. Zwietering¹, E.H.L. Aarts¹,² and J. Wessels¹,³

¹. Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands.
². Philips Research Laboratories, P.O. Box 80.000, 5600 JA Eindhoven, the Netherlands.

Abstract

In this paper we consider the problem of determining the minimal number of layers required by a multi-layered perceptron for solving the sorting problem. We discuss two formulations of the sorting problem; ABSSORT, which can be considered as the standard form of the sorting problem, and where, given an array of numbers, a new array with the original numbers in non-decreasing order is requested, and RELSORT, where, given an array of numbers, one wants to find the smallest number and for each number —except the largest— one wants to find the next largest number. We show that, if one uses classical multi-layered perceptrons with the hard-limiting response function, the minimal number of layers needed is 3 and 2 for solving ABSSORT and RELSORT, respectively.

Keywords: multi-layered perceptrons, minimal number of layers, neural networks, sorting
1 Introduction

An important issue in the design of feedforward neural networks is the choice of the number of layers. It is known that two-layered perceptrons can approximate any reasonable mapping with arbitrary precision if one uses a sufficiently large number of hidden units; see Cybenko [2], Funahashi [3] and Hornik, Stinchcombe & White [5]. As a special case it follows that every function representing a three-layered perceptron can be approximated with arbitrary precision by a two-layered perceptron. Therefore the approximative capabilities of two- and three-layered perceptrons are considered equivalent. However, if the number of hidden units is limited this equivalence is no longer true. This follows from the difference between the classification capabilities of two- and three-layered perceptrons that use the hard-limiting response function; see Gibson & Cowan [4], Huang & Lippmann [6], Li [8], Lippmann [9], Makhoul, Schwartz & El-Jaroudi [10], Wieland & Leighton [13] and Zwietering, Aarts & Wessels [14, 15]. The idea is that the existence of a hard-limiting \( m \)-layered perceptron with \( k \) hidden units that solves a given problem is a necessary condition for the existence of a class of sigmoid \( m \)-layered perceptrons with \( k \) hidden units that approximate the problem with arbitrary precision, for some fixed \( m \) and \( k \).

In this paper we demonstrate the difference between the capabilities of hard-limiting two- and three-layered perceptrons by applying them to different formulations of the sorting problem. Over the years the sorting problem has served as test-bed for new computing paradigms. Numerous sequential and parallel algorithms and circuits have been designed to solve this problem, each of which provides information about its sequential and parallel time and space complexity. For this reason, we use the sorting problem for examining the neural network model as a new massively parallel computing paradigm and for obtaining information about the neural complexity of sorting.

In a previous paper we showed that the sorting problem can be solved by a three-layered perceptron with an exponential number of hidden units; see [14]. However, the corresponding network was obtained by a general construction which does not guarantee any optimality, neither with respect to the required number of hidden units, nor with respect to the required number of layers. In a recent paper, Chen and Hsieh describe a feedforward neural network solution to the standard sorting problem that uses \( O(n^2) \) hidden units and 5 layers; see Chen & Hsieh [1]. Although this solution has a polynomial number of hidden units, the number of required layers is not minimal. Furthermore, their somewhat ad hoc approach uses several response functions, unbounded weights and can only sort numbers of equal sign.

In this paper we discuss two formulations of the sorting problem and the corresponding solution with a multi-layered perceptron. The first formulation, which can be considered as the standard formulation of the sorting problem, is ABSSORT, where, given an array of
n numbers, one wants to find a new array with the original numbers in non-decreasing order. We prove that three layers is the minimum for solving ABSSORT by a multi-layered perceptron. This is done by presenting a three-layered perceptron with \( O(n^2) \) hidden units that solves ABSSORT and proving that ABSSORT cannot be solved by a two-layered perceptron, whatever the size of the first hidden layer. The second formulation discussed is RELSORT, where, given an array of numbers, one wants to find the smallest number and for each number except the largest number one wants to find the next largest number. We prove that RELSORT can be solved by a two-layered perceptron with \( O(n^2) \) hidden units, which is again minimal with respect to the number of layers. Both the presented multi-layered perceptrons that solve ABSSORT and RELSORT have \( \frac{1}{2}n(n - 1) \) units in the first hidden layer and a total of \( O(n^2) \) units. It can be shown that the number of \( \frac{1}{2}n(n - 1) \) units in the first hidden layer is minimal. However, since the proof of this result falls outside the scope of this paper it is left out (cf. [16]).

The paper is organized as follows. Section 2 introduces the type of neural networks used in this paper, formalizes the considered problems and gives some preliminary results. In Section 3 the main results are presented. In Section 3.1 we prove that ABSSORT can be solved by a three-layered perceptron, in Section 3.2 we prove that ABSSORT cannot be solved by a two-layered perceptron and in Section 3.3 we prove that RELSORT can be solved by a two-layered perceptron, respectively. The paper ends with some concluding remarks and references.

## 2 Preliminaries

In this paper we consider the standard multi-layered perceptron architecture; see also Rumelhart, Hinton & Williams [12]. An \( m \)-layered perceptron (\( m \)-LP for short) consists of one output layer and \( m - 1 \) hidden layers. Every layer can have a different number of units and there are weighted connections only between units in subsequent layers. The output of a node is determined by a computation consisting of a summation of the bias and the weighted inputs of that node and passing the result passed through the hard-limiting response function \( \theta \). The output of a node is thus given by \( \theta(\sum_i a_i x_i + b) \), where \( x_i, a_i, b \in \mathbb{R} \) are the inputs, weights and bias, respectively and \( \theta \) is defined by:

\[
\theta(\lambda) = \begin{cases}
1 & \text{if } \lambda \geq 0, \\
0 & \text{if } \lambda < 0.
\end{cases}
\]  
(1)

Hence, an \( m \)-LP with \( n \) inputs and one output can be represented by a function \( f : \mathbb{R}^n \to \{0, 1\} \) and solves a given classification problem \((\mathbb{R}^n, \{V, V^\ast\})\) for some \( V \subseteq \mathbb{R}^n \), if it satisfies \( f(x) = 1 \) for all \( x \in V \) and \( f(x) = 0 \) for all \( x \in V^\ast = \mathbb{R}^n \setminus V \); for a discussion of the classification capabilities of \( m \)-LPs we refer to [14, 15] by the present authors.
The main issue of this paper concerns the existence and construction of an m-LP for the sorting problem. Sorting is the problem of finding a non-decreasing ordering of a given array of \( n \) real-valued numbers. Instead of presenting the requested ordering by a sorted array of the original numbers, we use indirect addressing by presenting an array of the indices of the sorted numbers. Since such an array can be viewed as a one-to-one mapping from \( \{1, \ldots, n\} \) to \( \{1, \ldots, n\} \), we use a permutation to denote such an array. If we assume for a moment that the given numbers are all distinct, then the solution to the problem is unique, in the sense that there exists only one non-decreasing ordering. We distinguish between the following two formulations of the sorting problem, that differ in the way a solution to the sorting problem is presented.

**Formulation 1 (ABSSORT)** Given an array of \( n \) real numbers, the problem is to find the absolute sequence in which the numbers have to be placed in order to obtain a sorted list, i.e., give the index of the smallest number, give the index of the one but smallest number, etc. Let \( \pi(i) \) denote the index of the number that takes position \( i \) in the sorted list. Then the problem can be formulated mathematically as:

- Given \( x_1, \ldots, x_n \in \mathbb{R} \),
- Find a mapping \( \pi : \{1, \ldots, n\} \to \{1, \ldots, n\} \) such that
  
  (i) \( \pi \) is a permutation,
  (ii) \( x_{\pi(i)} \leq x_{\pi(i+1)} \) for all \( i = 1, \ldots, n - 1 \).

**Formulation 2 (RELSORT)** Given an array of \( n \) real numbers, the problem is to find the relative sequence in which the numbers have to be placed in order to obtain a sorted list, i.e., give the index of the smallest number and for each number, except the largest, give the index of the number that follows this number in the sorted list. Let \( s \) denote the index of the smallest number and \( \alpha(i) \) denote the index of the number that is the successor of number \( i \) in the sorted list. Then the problem can be formulated mathematically as:

- Given \( x_1, \ldots, x_n \in \mathbb{R} \),
- Find \( s \in \{1, \ldots, n\} \) such that \( x_s \leq x_i \) for all \( i = 1, \ldots, n \), and a mapping \( \alpha : \{1, \ldots, n\} \to \{1, \ldots, n\} \) that satisfies
  
  (i) \( \alpha \) is a cyclic permutation,
  (ii) \( x_i \leq x_{\alpha(i)} \) for all \( i = 1, \ldots, n, i \neq l \),
  where \( l \in \{1, \ldots, n\} \) is such that \( x_l \leq x_i \) for all \( i = 1, \ldots, n \).

Note that a permutation \( \alpha \) is cyclic if \( \{\alpha^0(s), \alpha^1(s), \ldots, \alpha^{n-1}(s)\} = \{1, \ldots, n\} \). The condition that \( \alpha \) is a cyclic permutation is necessary, since there are permutations like...
\( \hat{\alpha}(i) = i \) for all \( i = 1, \ldots, n \) or \( \hat{\alpha}(s) = l, \hat{\alpha}(l) = s \) and \( \hat{\alpha}(i) = i \) for all \( i \neq s, l \), that satisfy all other conditions, but which are not the solution we have in mind.

Above we assumed for a moment that all numbers \( x_1, \ldots, x_n \) were distinct. This was necessary because otherwise the definitions of \( \pi, s \) and \( \alpha \) are ambiguous. However, in order to be able to solve a problem with an \( m \)-LP the solution has to be defined unambiguously; see also [14]. A formal way to treat the situation with possibly equal numbers in the array to be ordered, is to define the ordering \( \leq \) on the numbers \( x_1, \ldots, x_n \) as is done in the following lemma, where equal numbers are ordered according to their index value. Since the proof of the lemma is straightforward it is omitted.

**Lemma 1** If the ordering \( \leq \) on \( \{x_1, \ldots, x_n\} \) is defined as
\[
\quad x_i \leq x_j \equiv (x_i < x_j) \lor (x_i = x_j \land i < j) \lor (i = j),
\]
then the solutions of \textsc{abssort} and \textsc{relsort} are unique.

In the rest of the paper we use the ordering given by (2). One can use the result of Lemma 1 to show that the solutions of \textsc{abssort} and \textsc{relsort} are related.

**Corollary 1** The solutions to \textsc{abssort} and \textsc{relsort} are related as follows:

\[
\textsc{relsort} \to \textsc{abssort}: \pi(i) = \alpha^{-1}(s), \ i = 1, \ldots, n.
\]

\[
\textsc{abssort} \to \textsc{relsort}: s = \pi(1), \ l = \pi(n), \ \alpha(i) = \pi(\pi^{-1}(i) + 1), \ i \neq l \text{ and } \alpha(l) = s.
\]

### 3. Main Results

In this section we show that \textsc{abssort} can be solved by a 3-LP but cannot be solved by a 2-LP, which shows that three layers is minimal for \textsc{abssort}. Furthermore, we show that \textsc{relsort} can be solved by a 2-LP and, since it cannot be solved by a 1-LP, this is also minimal. By solving we mean that there exists an \( m \)-LP, such that for each array of \( n \) numbers \( x_1, \ldots, x_n \in \mathbb{R}^n \), given as inputs to the \( m \)-LP, its output is the solution of \textsc{abssort}, respectively \textsc{relsort}, for this array of numbers. Since we use hard-limiting response functions, we use a 0-1-representation for the solutions of \textsc{abssort} and \textsc{relsort}. To this end, we introduce two 0-1-matrices \( y, w \in \{0,1\}^{n \times n} \) and a 0-1-vector \( u \in \{0,1\}^n \) to represent \( \pi, \alpha \) and \( s \), the solutions of \textsc{abssort} and \textsc{relsort}, respectively: \( y_{ij} = 1_{\pi(i) = j}, w_{ij} = 1_{\alpha(i) = j} \) and \( u_i = 1_{s = i}. \)

Obviously \( y, w \) and \( s \) depend on the numbers \( x_1, \ldots, x_n \). Wherever needed, this functionality is written explicitly as \( y(x), w(x) \) and \( s(x) \), where \( x \) denotes \( (x_1, \ldots, x_n) \).

\(^1\)Here we use \( 1_{\{\cdot\}} \) to denote the true-false indicator: \( 1_{\text{true}} = 1, 1_{\text{false}} = 0. \)
3.1 A 3-LP for ABSSORT

In this section we prove that there exists a 3-LP represented by the function \( f(x) \) that solves ABSSORT. The first step is to give a reformulation of ABSSORT, in which Conditions (i) and (ii) given in the original formulation are combined.

**Lemma 2** Let \( x_1, \ldots, x_n \in \mathbb{R} \) and let the mapping \( \kappa : \{1, \ldots, n\} \to \{1, \ldots, n\} \) be defined by
\[
\kappa(j) = |\{k \in \{1, \ldots, n\} \mid x_k \leq x_j\}|,
\]
for all \( j = 1, \ldots, n \). Then \( \kappa \) is a permutation and \( \kappa^{-1} \) solves ABSSORT.

**Proof**
That \( \kappa \) defined by (3) is a permutation follows straightforwardly. Furthermore, \( \kappa(j) \) gives the position of \( x_j \) in the sorted list. This is equivalent to saying that \( \kappa^{-1}(i) \) is the index of the number that takes position \( i \) in the sorted list, which implies that \( \kappa^{-1} \) solves ABSSORT.

Next, we show that ABSSORT can be solved by a 3-LP, if one uses the formulation given by Lemma 2. We therefore note that for all \( j = 1, \ldots, n \) we have
\[
\pi^{-1}(j) = \kappa(j) = \sum_{k=1}^{n} h_{kj}(x),
\]
where the functions \( h_{ij}(x) \) are given by
\[
h_{ij}(x) = \begin{cases} 1 & \text{if } x_i \leq x_j, \\ 0 & \text{otherwise}, \end{cases}
\]
for all \( i, j \in \{1, \ldots, n\} \). From (2) and (1) we conclude that the functions given by (5) satisfy
\[
h_{ij}(x) = \begin{cases} \theta(x_j - x_i) & \text{if } i < j, \\ 1 & \text{if } i = j, \\ 1 - \theta(x_i - x_j) & \text{if } i > j, \end{cases}
\]
which shows that these functions essentially represent 1-LPs. This is the basic idea for the following construction of a 3-LP for \( y \), which is the 0-1-representation of \( \pi \), the solution of ABSSORT.

\[
y_{ij}(x) = 1 \iff \pi(i) = j
\]
\[
\iff \kappa(j) = i
\]
\[
\iff \sum_{k=1}^{n} h_{kj}(x) = i
\]
\[
\iff \sum_{k=1}^{n} h_{kj}(x) \geq i \land \sum_{k=1}^{n} h_{kj}(x) < i + 1
\]
Using (6) it follows that
\[ \sum_{k=1}^{n} h_{kj}(x) = \sum_{k=1}^{n-j} \theta(x_j - x_k) - \sum_{k=j+1}^{n} \theta(x_k - x_j) + n + 1 - j, \]
which implies that the functions \( g_{ij}(x) = \theta(\sum_{k=1}^{n} h_{kj} - i) \), \( i, j = 1, \ldots, n \), represent 2-LPs. Let \( g_{n+1,j}(x) \equiv 0 \) for all \( j \), then using (7) we find
\[ y_{ij}(x) = \theta(g_{ij}(x) - g_{i+1,j}(x) - 1), \]
for all \( i, j = 1, \ldots, n \), which proves the following theorem.

**Theorem 1** ABSSORT can be solved by a 3-LP with \( \frac{1}{2} n(n - 1) \) units in the first hidden layer, \( n^2 \) units in the second hidden layer and \( n^2 \) output units.

The basic idea behind the above given construction of a 3-LP for ABSSORT is the same idea as used by Chen and Hsieh in [1] for their construction of a 5-LP that solves the sorting problem. It is also the same idea as used by Preparata for his well-known parallel sorting algorithm that uses \( O(\log n) \) time and \( O(n^2) \) processors; see Preparata [11] and Kronsjo [7].

### 3.2 No 2-LP for ABSSORT

In this section we prove that ABSSORT cannot be solved by a 2-LP. We start by considering the case \( n = 3 \).

Suppose there exists a 2-LP that solves ABSSORT. Without loss of generality we assume that there exists an output unit \( z \), satisfying \( z = 1 \) if and only \( \pi(2) = 2 \). Then \( z = 1 \) if and only if \( x \in V \), where \( V \) is given by
\[ V = \{ x \in \mathbb{R}^3 \mid x_1 \leq x_2 \leq x_3 \lor x_3 \leq x_2 \leq x_1 \}. \]

A two-dimensional cut of \( V \) is shown in Figure 1. In other words there exists a 2-LP with one output, namely the part of the 2-LP that corresponds with the output \( z \), that solves the classification problem \( (\mathbb{R}^3, \{V, V^*\}) \). We complete our argument by showing that this is impossible. To this end we use Lemma 3 below, which gives a sufficient condition for a classification problem to be unsolvable by a 2-LP. We formulated and proved this lemma in [15]. For strongly related results see Gibson & Cowan [4]. The condition requires the existence of two spheres and a half-space. The sphere \( B(x_0, \delta) \) with center \( x_0 \in \mathbb{R}^n \) and radius \( \delta > 0 \) denotes the set \( \{ x \in \mathbb{R}^n \mid \| x - x_0 \| < \delta \} \). The half-space \( W(a, b) \) with
Figure 1: The set \( \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 \leq x_2 \leq x_3 \lor x_3 \leq x_2 \leq x_1 \} \) for a given value of \( x_3 \in \mathbb{R} \).

\( a \in \mathbb{R}^n \) and \( b \in \mathbb{R} \) denotes the set \( \{x \in \mathbb{R}^n \mid a \cdot x + b \geq 0\} \). If \( W = W(a, b) \) is a half-space, then the interior \( W^o \) and complement \( W^\ast \) of \( W \) are given by the (open) half-spaces \( W^o = \{x \mid a \cdot x + b > 0\} \) and \( W^\ast = \{x \mid a \cdot x + b < 0\} \), respectively.

**Lemma 3 ([4, 15])** Let \( V \) be a subset of \( \mathbb{R}^n \) for which there exist two spheres \( B_1, B_2 \) and a closed linear half-space \( W \) such that:

\[
\begin{align*}
\emptyset & \neq B_1 \cap W^o \subseteq V \\
\emptyset & \neq B_1 \cap W^\ast \subseteq V^\ast
\end{align*}
\]

then the classification problem \( (\mathbb{R}^n, \{V, V^\ast\}) \) cannot be solved by a two-layered perceptron.

One can easily verify that (9) holds if \( V \) is given by (8), \( B_1 = B((1,1,2),1) \), \( B_2 = B((3,3,2),1) \) and \( W = W((-1,1,0),0) \), which proves that there does not exist a 2-LP for ABSSORT for \( n = 3 \).

For \( n > 3 \) exactly the same argument can be used as for \( n = 3 \); in this case (9) holds for \( V = \{x \in \mathbb{R}^n \mid \pi(2) = 2\} \), \( B_1 = B((1,1,2,4,\ldots,4),1) \), \( B_2 = B((3,3,2,4,\ldots,4),1) \) and \( W = W((-1,1,0,0,\ldots,0),0) \). This completes the proof of the following theorem.

**Theorem 2** There does not exist a 2-LP that solves ABSSORT.
### 3.3 A 2-LP for RELSORT

In this section we prove that there exist two 2-LPs represented by the functions \( f(x) \) and \( g(x) \) such that \( w(x) = f(x) \) and \( s(x) = g(x) \) for all \( x \in \mathbb{R}^n \). Combined they form a 2-LP that solves RELSORT. We start by giving a reformulation of RELSORT. This reformulation is necessary since Condition (i), demanding that \( \alpha \) is a cyclic permutation, is a hard condition to verify in a distributed environment. In the following lemma we show that this condition can be replaced by a set of local constraints if we simultaneously strengthen condition (ii).

**Lemma 4** The conditions (i) and (ii) in the formulation of RELSORT can be replaced by the following set of conditions:

(i') \( \alpha(i) \neq i \) for all \( i \neq l \) and \( \alpha(l) = s \),

(ii') \( x_i \leq x_{\alpha(i)} \land (x_i \leq x_j \Rightarrow x_{\alpha(i)} \leq x_j) \) for all \( i \neq l, j \neq i \),

where \( l \in \{1, \ldots, n\} \) is such that \( x_i \leq x_l \) for all \( i = 1, \ldots, n \).

**Proof**

Let \( \alpha \) be a mapping from \( \{1, \ldots, n\} \) to \( \{1, \ldots, n\} \) satisfying (i') and (ii') given above. It is obvious that \( \alpha \) satisfies (ii) and hence it remains to prove that \( \alpha \) is a cyclic permutation. Since the proof is trivial for \( n = 1 \) we assume \( n \geq 2 \), which implies \( s \neq l \) and, hence, \( \alpha(s) \neq s \).

First, we show that \( \alpha(i) \neq \alpha(j) \) for all \( i \neq j \), which implies that \( \alpha \) is a permutation. Assume \( \alpha(i) = \alpha(j) \) for some \( i \neq j \). Without loss of generality we assume \( x_i \leq x_j \), which implies that \( i \neq l \).

If \( j = l \), then \( x_s \leq x_i \leq x_{\alpha(i)} = x_{\alpha(j)} = x_{\alpha(l)} = x_s \), contradicting \( \alpha(i) \neq i \).

If \( j \neq l \), then, since \( x_i \leq x_j \), we have \( x_{\alpha(j)} = x_{\alpha(i)} \leq x_j \leq x_{\alpha(j)} \), contradicting \( \alpha(j) \neq j \).

Next, we show that \( \{\alpha^0(s), \alpha^1(s), \ldots, \alpha^{n-1}(s)\} = \{1, \ldots, n\} \), which proves that \( \alpha \) is a cyclic permutation. Since \( \alpha \) is a permutation, one can easily argue that there exists a \( k \in \{1, \ldots, n-1\} \) such that \( \alpha^k(s) = l \) and \( \alpha^i(s) \neq l \) for all \( i = 0, \ldots, k-1 \). This implies

\[
x_s = x_{\alpha^0(s)} \leq x_{\alpha^1(s)} \leq \cdots \leq x_{\alpha^{k-1}(s)} \leq x_{\alpha^k(s)} = x_l.
\]

(10)

Let \( j \in \{1, \ldots, n\} \), then from (10) we conclude that \( x_{\alpha^{i-1}(s)} \leq x_j \leq x_{\alpha^i(s)} \) for some \( i \in \{1, \ldots, k\} \). If \( j \neq \alpha^{i-1}(s) \), then, using \( \alpha^{i-1}(s) \neq l \), it follows that \( x_{\alpha^i(s)} \leq x_j \leq x_{\alpha^i(s)} \), which implies that \( j = \alpha^i(s) \).

Next, we show that RELSORT can be solved by a 2-LP, if one uses the conditions given by Lemma 4. First, we consider the case \( n = 3 \), let \( x = (x_1, x_2, x_3) \in \mathbb{R}^3 \).
That \( u(x) \) can be solved by a 2-LP follows straightforwardly:

\[
\begin{align*}
  u_i(x) = 1 & \iff s = i \\
  & \iff x_i \leq x_1 \land x_i \leq x_2 \land x_i \leq x_3 \\
  & \iff h_{i1}(x) \geq 1 \land h_{i2}(x) \geq 1 \land h_{i3}(x) \geq 1 \\
  & \iff h_{i1}(x) + h_{i2}(x) + h_{i3}(x) \geq 3,
\end{align*}
\]

where the functions \( h_{ij}(x) \) are given by (5) in Section 3.1. Hence, using (6), it follows that

\[
\begin{align*}
  u_1(x) &= \theta(\theta(x_2 - x_1) + \theta(x_3 - x_1) - 2), \\
  u_2(x) &= \theta(-\theta(x_2 - x_1) + \theta(x_3 - x_2) - 1), \\
  u_3(x) &= \theta(-\theta(x_3 - x_1) - \theta(x_3 - x_2)),
\end{align*}
\]

which completes the construction of a 2-LP for \( u \). It remains to show that \( w \) can be solved by a 2-LP.

First, since \( \alpha(i) \neq i \) for \( i = 1, 2, 3 \), we have \( w_{11}(x) = w_{22}(x) = w_{33}(x) = 0 \). Next, consider \( w_{12} \). We have \( w_{12} = 1 \) if and only if \( \alpha(1) = 2 \).

If \( l \neq 1 \), then \( \alpha(1) = 2 \) if and only if \( x_1 \leq x_2 \leq x_3 \) or \( x_3 \leq x_1 \leq x_2 \).

If \( l = 1 \), then \( s = \alpha(l) = \alpha(1) = 2 \). Furthermore, \( l = 1 \) and \( s = 2 \) if and only if \( x_2 \leq x_3 \leq x_1 \).

Hence, \( w_{12} = 1 \) if and only if

\[
(x_1 \leq x_2 \leq x_3) \lor (x_3 \leq x_1 \leq x_2) \lor (x_2 \leq x_3 \leq x_1). \quad (11)
\]

For a given \( x_3 \in \mathbb{R} \) the subset of \( \mathbb{R}^2 \) defined by (11) is depicted in Figure 2. Proving that \( w_{12} \) can be solved by a 2-LP is equivalent to proving that the subset given in Figure 2 can be classified by a 2-LP for all \( x_3 \in \mathbb{R} \).

One can easily show that (11) is equivalent to

\[
h_{31}(x) + h_{12}(x) + h_{23}(x) \geq 2, \quad (12)
\]

where the functions \( h_{ij}(x) \) are given by (5), in Section 3.1. Hence, using (6), it follows that

\[
w_{12}(x) = \theta(-\theta(x_3 - x_1) + \theta(x_2 - x_1) + \theta(x_3 - x_2) - 1),
\]

which proves that \( w_{12}(x) \) can be solved by a 2-LP. Similarly, one can show that \( w_{23}(x) = w_{31}(x) = w_{12}(x) \) and

\[
w_{13}(x) = w_{21}(x) = w_{32}(x) = \theta(h_{21}(x) + h_{13}(x) + h_{32}(x) - 2)
  = \theta(-\theta(x_2 - x_1) + \theta(x_3 - x_1) - \theta(x_3 - x_2)),
\]

9
Figure 2: The set \((x_1, x_2) \in \mathbb{R}^2 \mid x_1 \leq x_2 \leq x_3 \lor x_3 \leq x_1 \leq x_2 \lor x_2 \leq x_3 \leq x_1\) for a given value of \(x_3 \in \mathbb{R}\).

hereby completing the construction of a 2-LP for \(w\).

Next, we consider the general case \(n \geq 3\). In general the construction of a 2-LP for \(u\) is not harder than for \(n = 3\). We find that

\[ u_i(x) = \theta(\sum_{k=1}^{n} h_{ik}(x) - n), \]

which can be shown to represent a 2-LP using (6), similarly as was done for the functions \(g_{ij}(x)\) that were introduced in Section 3.1.

The construction of a 2-LP that solves \(w\) in the general case, starts by noting that for all \(x \in \mathbb{R}^n\) and \(i, j, k = 1, \ldots, n, i \neq j\)

\[ h_{ki}(x) + h_{ij}(x) + h_{jk}(x) \leq 2. \]  \hspace{1cm} (13)

This will enable us to use the same approach for \(n > 3\) as we used for \(n = 3\).

Again we have \(w_{ii}(x) = 0\) for all \(i = 1, \ldots, n\). For \(i \neq j\) we find

\[ w_{ij}(x) = 1 \iff \alpha(i) = j \]
\[ [(x_i \leq x_j) \land (\forall k : x_k \leq x_i \lor x_j \leq x_k)] \]
\[ \lor [(x_j \leq x_i) \land (\forall k : x_j \leq x_k \leq x_i)] \]
\[ \forall k : (x_k \leq x_i \leq x_j) \lor (x_i \leq x_j \leq x_k) \lor (x_j \leq x_k \leq x_i) \]
\[ \forall k : h_{ki}(x) + h_{ij}(x) + h_{jk}(x) \geq 2 \]
\[ \sum_{k=1}^{n} h_{ki}(x) + nh_{ij}(x) + \sum_{k=1}^{n} h_{jk}(x) \geq 2n, \]

where we used (13) in the last step. This proves that \( w(x) \) can be solved by a 2-LP and, hence, completes the proof of the following theorem.

**Theorem 3** RELSORT can be solved by a 2-LP with \( \frac{1}{2}n(n - 1) \) units in the first hidden layer and \( n(n + 1) \) output units.

In the above theorem \( n(n + 1) \) output units are used, \( n \) for \( u \) and \( n^2 \) for \( w \). However, since the units \( w_{ii} \) (\( i = 1, \ldots, n \)) are identical to zero, they can be left out in order to reduce the total number of output units to \( n^2 \).

The fact that a 2-LP can be found that solves RELSORT in the general case, is due to property (13), which in its turn is due to the addition of the constraint \( \alpha(l) = s \); see also Figure 2, where the addition of \( \alpha(l) = s \) corresponds to the addition of the bottom-right part, which makes the total figure symmetrical. Hence, although the constraint \( \alpha(l) = s \) is not necessary for the problem formulation, it turns out to be crucial for the solution by a 2-LP.

## 4 Concluding Remarks

This paper discussed the following question: what is the minimal number of layers that a multi-layered perceptron must have for solving the sorting problem. We discussed two formulations of the sorting problem: ABSSORT, which can be considered as the standard sorting problem, and RELSORT, where, given an array of numbers, one wants to find the smallest number, and for each number, except the largest, one wants to find the next largest number. We showed that ABSSORT and RELSORT can be solved by a three-layered perceptron (3-LP) and two-layered perceptron (2-LP), respectively, and that this is minimal with respect to the number of layers. Both the presented \( m \)-LPs have \( n \) inputs, \( O(n^2) \) hidden units and \( n^2 \) output units. In the introduction of this paper we stated that the solutions of ABSSORT and RELSORT are related. Therefore, one might wonder whether there exist a 1-LP, which, if put on top of the 2-LP that solves RELSORT, yields a 3-LP that solves ABSSORT. The answer is negative, as one can straightforwardly show that the posed question leads to a classification problem that is not separable.
We considered the classical $m$-LP-architecture. One implication is that we assumed the inputs to be real-valued numbers. Therefore, the discussed sorting problems are also defined for an array of real-valued numbers and the minimality results derived in this paper are valid in this situation only. If the numbers that have to be sorted are for instance integer and bounded, then there exists a $2$-LP that solves ABSSORT. This follows from the observation that the intersection of the set depicted in Figure 1 with the set $\mathbb{Z}_k^2 = \{x \in \mathbb{Z}^2 | -k \leq x_i \leq k, i = 1, 2\}$ (for some $k \in \mathbb{N}$) can be classified by a $2$-LP. This is done by embedding this set in an appropriate subset of $\mathbb{R}^2$, see Figure 3 and also Makhoul, Schwartz & El-Jaroudi [10] and Zwietering, Aarts & Wessels [15]. Based on

![Figure 3: Embedding the set \{(x_1, x_2) \in \mathbb{Z}^2 | x_1 \leq x_2 \leq x_3 \lor x_3 \leq x_2 \leq x_1\}, for a given value of $x_3 \in \mathbb{Z}$, in a subset of $\mathbb{R}^2$ that can be classified by a two-layered perceptron.](image)

results found for $n = 3$ we expect that the bounded integer sorting problem can be solved by a $2$-LP with $O(k \cdot n!)$ hidden units in general. Since the $3$-LP presented in this paper solves the same problem with $O(n^2)$ hidden units this extension supports our basic results about the difference between the capabilities of two- and three-layered perceptrons.

A second implication of our choice for classical $m$-LPs is that we allow connections between units in subsequent layers only. This, combined with the use of the hard-limiting response function can be shown to imply that $\frac{1}{2}n(n - 1)$ is the minimal number of first layer units that is required by an $m$-LP for solving ABSSORT and RELSORT; see Zwietering, Aarts
& Wessels [16]. This implies that the 3-LP and 2-LP presented in this paper for solving ABSSORT and RELSORT, are also minimal with respect to the number of units in the first hidden layer. If connections between units in non-subsequent layers are admitted, then it is not hard to construct a $\mathcal{O}(\log^2 n)$-LP with $\mathcal{O}(n)$ units in every hidden layer that solves ABSSORT, using the principles of so-called comparator networks; see also Kronsjö [7].

References


