Journal of Machine Learning Research 5 (2004) 891–912 Submitted 3/04; Published 8/04

Some Dichotomy Theorems for Neural Learning Problems

Michael Schmitt MSCHMITT@LMI.RUHR-UNI-BOCHUM.DE
Lehrstuhl Mathematik und Informatik
Fakultät für Mathematik
Ruhr-Universität Bochum
D–44780 Bochum, Germany

Editor: Manfred K. Warmuth

Abstract

The computational complexity of learning from binary examples is investigated for linear threshold neurons. We introduce combinatorial measures that create classes of infinitely many learning problems with sample restrictions. We analyze how the complexity of these problems depends on the values for the measures. The results are established as dichotomy theorems showing that each problem is either NP-complete or solvable in polynomial time. In particular, we consider consistency and maximum consistency problems for neurons with binary weights, and maximum consistency problems for neurons with arbitrary weights. We determine for each problem class the dividing line between the NP-complete and polynomial-time solvable problems. Moreover, all efficiently solvable problems are shown to have constructive algorithms that require no more than linear time on a random access machine model. Similar dichotomies are exhibited for neurons with bounded threshold. The results demonstrate on the one hand that the consideration of sample constraints can lead to the discovery of new efficient algorithms for non-trivial learning problems. On the other hand, hard learning problems may remain intractable even for severely restricted samples.

Keywords: linear threshold neuron, consistency problem, computational complexity, linear-time algorithm, NP-completeness

1. Introduction

The ability to learn is certainly one of the most challenging qualities that people have ever demanded from computers. There is no doubt that a major advancement in the development of learning algorithms has been made due to the use of neural networks. Today, they provide the basis for some of the most successful learning techniques. This achievement, however, is compromised by the fact that the running times required by neural learning algorithms are often immense. On the theoretical side, investigations of the computational complexity of learning problems have supported the conjecture that neural learning is generally hard: Theory considers an algorithm as efficient if its running time is bounded by a polynomial in the input length. Based on the theory of NP-completeness (see, e.g., Garey and Johnson, 1979) researchers have established numerous intractability results for neural learning problems. Consequently, no efficient, that is, polynomial-time, algorithm for solving these problems can exist if the complexity classes P and NP are different. (See, e.g., Šíma, 2002, for a comprehensive list of hardness results concerning single neurons and neural networks.)

A useful approach for studying the complexity of learning is to consider the consistency problem. Associated with a class of functions, the so-called hypothesis class, the consistency problem is a decision problem. It poses the question whether for a given set of labeled examples, the training sample, there is some hypothesis in the class that is consistent with all examples, that is, assigns the correct output values. The consistency problem is also known as loading problem (Judd, 1990), fitting problem (Natarajan, 1991), or training problem (Blum and Rivest, 1992). A variant of the consistency problem is the maximum consistency problem, also known as minimizing disagreement problem. Its instances consist of a training sample and a natural number k. The question is to decide whether there is a hypothesis consistent with some subset of at least k examples. The maximum consistency problem is a generalization of the consistency problem in that the latter is a subproblem of the former: The consistency problem can be obtained from the maximum consistency problem by setting k equal to the number of examples. Consequently, if for a given hypothesis class the consistency problem is NP-hard then the maximum consistency problem is NP-hard, too. Consistency and maximum consistency problems are studied as key problems in the computational models of learning known as probably approximately correct (PAC) learning and agnostic PAC learning. A fundamental result states that, under the assumption that the complexity classes RP and NP are different, PAC learning cannot be efficient for a hypothesis class if the consistency problem for this class is NP-hard (Pitt and Valiant, 1988; Blumer et al., 1989). An analogous result links the model of agnostic PAC learning with the maximum consistency problem (Kearns et al., 1994; Anthony and Bartlett, 1999).

There has been perseverating interest in theoretical issues of learning using single neurons as hypothesis class (see, e.g., Servedio, 2002, and the references there). One of the extensively studied models is the linear threshold neuron, also known as McCulloch-Pitts neuron (McCulloch and Pitts, 1943). It serves as elementary building block for many neural network types (see, e.g., Haykin, 1999). While the linear threshold neuron has arbitrary real-valued weights, a variant that has gained particular attention is obtained by restricting the weights to binary values (see, e.g., Golea and Marchand, 1993; Fang and Venkatesh, 1996; Kim and Roche, 1998; Sommer and Palm, 1999). With binary weights a linear threshold neuron is still able to compute fundamental Boolean functions such as conjunction, disjunction, and r-of-k threshold functions[1] that have been of specific interest in questions of learning (Hampson and Volper, 1986; Littlestone, 1988; Pitt and Valiant, 1988).

In this article, we investigate the complexity of consistency and maximum consistency problems for linear threshold neurons with binary and with arbitrary weights. We do this under the assumption that the training examples are binary, that is, have entries from $\{0,1\}$. While the consistency problem for linear threshold neurons with arbitrary weights is known to be solvable in polynomial time using methods of linear programming (see, e.g., Maass and Turán, 1994; Anthony and Bartlett, 1999) we focus on those problems that are NP-complete: The consistency problem for linear threshold neurons with binary weights has been proved to be NP-complete by Pitt and Valiant (1988). Hence, also the maximum consistency problem for these neurons is NP-complete. The maximum consistency problem for linear threshold neurons with arbitrary weights is known to be NP-complete from the work of Höffgen et al. (1995) improving an earlier result by Amaldi (1991).

We take a closer look at these problems by introducing and analyzing subproblems. In other words, we raise the question whether these problems become easier when the training samples are restricted. To this end we define two combinatorial measures for characterizing the complexity of a sample. The first quantity, called coincidence, indicates the maximum number of components in which any two examples are both non-zero. The second measure, called sparseness, represents the maximum number of non-zero components, or the Hamming-weight, of any example. The

1. A Boolean r-of-k threshold function outputs 1 if and only if at least r in a specified subset of k variables have the value 1.

choice of these measures is guided by the insight that learning for single neurons is easy when the coincidence or the sparseness of the samples is severely limited. So, it is not hard to see that the consistency problem is trivial when the examples are pairwise orthogonal, that is, when the sample has coincidence zero. The same observation can be made for the case when the examples are required to be extremely sparse. A further motivation for introducing sparseness arises from investigations of neural associative memories. In these, sparseness has been shown to be relevant for their storage capacities using specific learning rules (Palm, 1980).

By introducing coincidence and sparseness as sample restrictions we obtain infinite classes of consistency and maximum consistency problems: For each pair c, d of natural numbers there is the (maximum) consistency problem with coincidence c and sparseness d. The question arises how the complexity of the consistency and the maximum consistency problems depends on the values for coincidence and sparseness. As one of the main results it emerges that these problems are already NP-complete when coincidence and sparseness are bounded, a fact that does not follow from previous work. Moreover, we give a complete categorization of these consistency and maximum consistency problems into NP-complete and polynomial-time solvable. In other words, we establish so-called dichotomy theorems for these problems. Given an infinite class of problems, a dichotomy theorem states that each problem is either NP-complete or contained in the complexity class P. Thus, if P $\neq$ NP, the dichotomy theorem separates the efficiently solvable problems from the intractable ones. A dichotomy theorem is an important finding since it is known that if P and NP are different then there are infinitely many problems in NP that are neither in P nor NP-complete (see, e.g., Garey and Johnson, 1979). It is therefore significant that none of these problems of intermediate complexity is found among the (maximum) consistency problems considered here. In computational complexity there are only a few dichotomy theorems known. The most popular one is that for k-SAT which states that the satisfiability problem for conjunctions of Boolean clauses, where each clause has size at most k, is NP-complete when k is at least 3 and solvable in polynomial time for k at most 2. The dichotomy of k-SAT emerges as a special case of the renowned result of Schaefer (1978) who established a more general dichotomy theorem for satisfiability problems. Concerning more recent results we refer to Kirousis and Kolaitis (2003) where also a brief survey on dichotomy theorems in computational complexity is given. The first dichotomy theorem in conjunction with learning problems has been provided by Dalmau (1999) for learning quantified Boolean formulas (see also Dalmau and Jeavons, 2003).

In detail, we prove the following dichotomy theorems: The consistency problem for the linear threshold neuron with binary weights is NP-complete if the samples have coincidence at least 1 and sparseness at least 4. On the other hand, if the samples have coincidence 0 or sparseness at most 3, the problem becomes solvable in polynomial time. Further, the maximum consistency problem for the linear threshold neuron with binary weights is NP-complete for coincidence at least 1 and sparseness at least 2, whereas for coincidence 0 or sparseness 1 it is in P. This last dichotomy theorem holds also for the maximum consistency problem with respect to the linear threshold neuron with arbitrary weights. All polynomial-time results are established by showing that the problems can be solved in linear time on a random access machine (RAM) model. Moreover, the algorithms we present for these problems are constructive, that is, they calculate a solution if one exists. We further obtain dichotomies for linear threshold neurons with binary weights and a bounded threshold. In particular, the dichotomy of the consistency problems that holds for these neurons with arbitrary threshold is also existent when the threshold is bounded by some value larger than or equal to 2. On the other hand, if the bound on the threshold is at most 1, the consistency problem is solvable in linear time on a RAM. For the maximum consistency problem and linear threshold neurons with binary weights NP-completeness is found at a value for the threshold of 1 or greater, whereas linear-time solvability holds when the threshold is equal to 0.

The dichotomy theorems presented here extend the NP-completeness results for learning with single neurons to an infinite class of subproblems. Therefore, they confirm that neural learning belongs to the hardest computational problems that one needs to solve in practice. On the other hand, those problems that we find to be solvable in polynomial time can be solved even in linear time on a RAM. The algorithms designed for these problems give interesting new insights. Moreover, efficiently solvable subproblems have been successfully used for the design of improved heuristics for NP-complete problems (Zheng and Stuckey, 2002). Therefore, it is reasonable to expect that the new linear-time algorithms presented here may lead to new and faster heuristics for learning in single neurons and neural networks.

In Section 2 we introduce the definitions of the basic concepts. The dichotomy theorems are stated in Section 3. The proofs are given in the following two sections: Section 4 is concerned with the NP-completeness results while the linear-time algorithms are presented in Section 5. Section 6 is devoted to some conclusions and open questions. In the appendix we establish the NP-completeness of a new variant of the satisfiability problem that is used in Section 4. We assume that the reader is familiar with the theory of NP-completeness as covered, for instance, by Garey and Johnson (1979).

Bibliographic Note. Some of the results in this article have been previously described in a technical report (Schmitt, 1994a) and presented at the International Conference on Artificial Neural Networks, ICANN '95, in Paris (Schmitt, 1995). This article is a thoroughly revised and extended version of these papers.

2. Neurons, Samples, Problems, and Algorithms

We begin by defining the neuron model and the function classes associated with it. Then we introduce the notion of a sample and specify the computational problems involved with neurons and samples. Finally, with regard to the linear-time algorithms presented in Section 5, we describe the computer model of the random access machine (RAM) on which the time bounds rely.

2.1 Neurons

A linear threshold neuron has parameters $w_1, \ldots, w_n, t \in \mathbb{R}$, where $w_1, \ldots, w_n$ are the weights and t is the threshold. The class of linearly separable Boolean functions in n variables, denoted $\mathcal{L}_n$, is the class of functions $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a linear threshold neuron. Specifically, for every $f \in \mathcal{L}_n$ there are weights $w_1, \ldots, w_n$ and a threshold t such that for all $x \in \{0,1\}^n$,

$$f(x) \quad = \quad \begin{cases} 1 & \text{if } w_1x_1 + \cdots + w_nx_n \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

We use $\mathcal{B}_n$ to denote the subclass of linearly separable Boolean functions in n variables that can be computed by a linear threshold neuron using only binary weights, that is, $w_1, \ldots, w_n \in \{0,1\}$. Further, $\mathcal{B}_n^\ell$ is the subclass of functions in $\mathcal{B}_n$ that can be computed with threshold $t \leq \ell$. Clearly, a value of $\ell \in \{0, \ldots, n\}$ is sufficient and we have $\mathcal{B}_n^n = \mathcal{B}_n$.

By omitting the subscript n we refer to the union of the respective classes such as, for instance, $\mathcal{L} = \bigcup_{n \geq 1} \mathcal{L}_n$.

2.2 Samples

A sample S is a finite set $S \subseteq \{0,1\}^n \times \{0,1\}$. The elements of S are called examples. Given a sample S, the set of positive examples $\mathrm{Pos}(S)$ and the set of negative examples $\mathrm{Neg}(S)$ of S are defined by

$$\begin{aligned}\mathrm{Pos}(S) &= \{x \mid (x,1) \in S\},\\ \mathrm{Neg}(S) &= \{x \mid (x,0) \in S\}.\end{aligned}$$

A sample may contain examples that are both positive and negative. Otherwise, that is, if $\mathrm{Pos}(S) \cap \mathrm{Neg}(S) = \emptyset$, the sample is said to be consistent.[2] The domain of a sample S, denoted $\mathrm{Dom}(S)$, is the set

$$\mathrm{Dom}(S) = \mathrm{Pos}(S) \cup \mathrm{Neg}(S).$$

We introduce two combinatorial measures that assign natural numbers to samples: coincidence and sparseness. The coincidence of a sample S, denoted $\mathrm{Coin}(S)$, is

$$\mathrm{Coin}(S) = \begin{cases} \max\{x \cdot y \mid x,y \in \mathrm{Dom}(S),\ x \neq y\} & \text{if } |S| \geq 2,\\ 0 & \text{otherwise,}\end{cases}$$

where $x \cdot y$ denotes the inner product. The sparseness of S, $\mathrm{Sparse}(S)$, is

$$\mathrm{Sparse}(S) = \begin{cases} \max\{x \cdot x \mid x \in \mathrm{Dom}(S)\} & \text{if } S \neq \emptyset,\\ 0 & \text{otherwise.}\end{cases}$$

Thus, $\mathrm{Coin}(S)$ is the maximum number of components in which two different elements in the domain of S have a 1 in common, and $\mathrm{Sparse}(S)$ is the maximum number of 1s, or the Hamming weight, of an element. Note that $\mathrm{Coin}(S) \leq \mathrm{Sparse}(S)$. Thus, bounding the sparseness of samples by some constant also bounds their coincidence by the same constant.

2.3 Consistency Problems

Given a sample S, a function f is said to be consistent with S if f satisfies $f(x) = a$ for all $(x,a) \in S$. The consistency problem for a class $\mathcal{F}$ of Boolean functions is the problem to decide whether for an input sample S there exists some function $f \in \mathcal{F}$ that is consistent with S. Loading problem, fitting problem, and training problem are synonyms for the consistency problem.

Let $\mathcal{F}$ be a class of Boolean functions and let $c,d \in \mathbb{N}$. We define the following consistency problems for $\mathcal{F}$ with sample restrictions:

$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c
Instance: A sample S with $\mathrm{Coin}(S) \leq c$.
Question: Is there a function $f \in \mathcal{F}$ that is consistent with S?

$\mathcal{F}$-CONSISTENCY WITH SPARSENESS d
Instance: A sample S with $\mathrm{Sparse}(S) \leq d$.
Question: Is there a function $f \in \mathcal{F}$ that is consistent with S?

2. The term “consistent sample” uses the word “consistent” in a somewhat different sense than the rest of the article.

The definition of the problem $\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d is similar.

The instances of the maximum consistency problem for a class $\mathcal{F}$ of functions consist of a sample S and a number $k \in \mathbb{N}$. The question is whether there exists a subset of S with at least k elements and a function $f \in \mathcal{F}$ consistent with that subset. Given numbers $c, d \in \mathbb{N}$, we define maximum consistency problems for $\mathcal{F}$ with sample restrictions as follows:

MAX-$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c
Instance: A sample S with $\mathrm{Coin}(S) \leq c$ and a natural number k.
Question: Is there a subsample $S' \subseteq S$ with $|S'| \geq k$ and a function $f \in \mathcal{F}$ that is consistent with S'?

MAX-$\mathcal{F}$-CONSISTENCY WITH SPARSENESS d
Instance: A sample S with $\mathrm{Sparse}(S) \leq d$ and a natural number k.
Question: Is there a subsample $S' \subseteq S$ with $|S'| \geq k$ and a function $f \in \mathcal{F}$ that is consistent with S'?

Further, we consider the problem called MAX-$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d, which is the problem where the instances satisfy both constraints.

There are some natural relationships among the complexities of these problems. Clearly, an algorithm that solves the consistency problem with sparseness d solves also every consistency problem with sparseness less than d for the same function class. Furthermore, if the consistency problem is shown to be NP-hard for some sparseness d then it follows that all consistency problems with sparseness larger than d for the same function class are NP-hard as well, since the reduction established for sparseness d is also valid for larger bounds on the sparseness.

Similar considerations can be made with regard to coincidence. Moreover, complexity results for consistency problems with bounded coincidence have consequences for the complexity of consistency problems with bounded sparseness, and vice versa. In particular, an algorithm that solves the consistency problem for coincidence c solves also the consistency problem for sparseness $c+1$ since every sample with sparseness at most $c+1$ has coincidence no more than c. (Note that coincidence is measured using non-identical pairs only.) Correspondingly, NP-hardness for sparseness d implies NP-hardness for coincidence $d-1$.

Analogous interdependences hold among the maximum consistency problems with sample restrictions. Additionally, maximum consistency problems are related to consistency problems in that the consistency problem is a subproblem of the corresponding maximum consistency problem. The former is obtained from the latter by letting the number k be equal to the cardinality of the sample.

2.4 Linear-Time Algorithms

We give linear-time algorithms for all problems not shown to be NP-complete. While the complexity class P is known to be widely machine independent, this is not the case for the class of problems solved in linear time on some machine. In fact, a detailed account of the computer model must be given. We adopt the model of a random access machine (RAM) as it is the most realistic formal computer model and frequently used when analyzing the running time of algorithms (see, e.g., van Emde Boas, 1990). The instruction set of the RAM model comprises the standard load and store operations including the use of indirect addresses. Its arithmetic operations are addition and subtraction, whereas multiplication and division are not available. We give time bounds for RAMs in the uniform measure, that is, the execution of an instruction counts as one unit of time. It is known that every RAM that requires time $t(n)$ in the uniform measure can be simulated on a multi-tape Turing machine in time $O(t^3(n))$ (Cook and Reckhow, 1973). In particular, a linear-time algorithm on a RAM ensures that the problem it solves belongs to P.

3. Dichotomy Theorems

We present five dichotomy theorems about the complexity of consistency problems with sample restrictions. They deal with the function classes $\mathcal{B}, \mathcal{B}^\ell$, and $\mathcal{L}$ introduced in Section 2.1 and the consistency problems and maximum consistency problems with sample restrictions defined in Section 2.3. The dichotomy theorems establish that, with one exception, each of these problems is either NP-complete or solvable in polynomial time. The latter is shown by exhibiting linear-time algorithms in all cases. The exception is the problem class arising from $\mathcal{L}$-CONSISTENCY. All subproblems of $\mathcal{L}$-CONSISTENCY can be solved in polynomial time as $\mathcal{L}$-CONSISTENCY itself is solvable in polynomial time using linear programming (see, e.g., Anthony and Bartlett, 1999; Maass and Turán, 1994). Therefore, if P $\neq$ NP, none of the subproblems of $\mathcal{L}$-CONSISTENCY is NP-complete.

The proofs for the statements that claim NP-completeness are presented in Section 4. The linear time bounds are established in Section 5 by giving algorithms that solve the problems constructively, that is, provide a solution if there exists one.

The first dichotomy theorem is concerned with the consistency problem for neurons with binary weights and arbitrary threshold.

Theorem 1 *$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d is NP-complete whenever $c \geq 1$ and $d \geq 4$. On the other hand, $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 and $\mathcal{B}$-CONSISTENCY WITH SPARSENESS d for $d \leq 3$ is solvable in linear time.*

An analogous result holds for neurons where the threshold is bounded by a constant of value at least 2. Moreover, if the threshold is required to be at most 1, we obtain solvability in linear time for all values of coincidence and sparseness.

Theorem 2 *For every $\ell \geq 2$, Theorem 1 holds with $\mathcal{B}^\ell$ in place of $\mathcal{B}$. On the other hand, $\mathcal{B}^1$-CONSISTENCY WITH COINCIDENCE c and $\mathcal{B}^1$-CONSISTENCY WITH SPARSENESS d is solvable in linear time for all c and d.*

These statements are complemented by the following two theorems that consider the corresponding maximum consistency problems.

Theorem 3 MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d *is NP-complete whenever $c \geq 1$ and $d \geq 2$. On the other hand,* MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{B}$-CONSISTENCY WITH SPARSENESS d *for $d \leq 1$ is solvable in linear time.*

A corresponding result for neurons with bounded threshold is the following one.

Theorem 4 *For every $\ell \geq 1$, Theorem 3 holds with $\mathcal{B}^\ell$ in place of $\mathcal{B}$. On the other hand,* MAX-$\mathcal{B}^0$-CONSISTENCY WITH COINCIDENCE c *and* MAX-$\mathcal{B}^0$-CONSISTENCY WITH SPARSENESS d *is solvable in linear time for all c and d.*

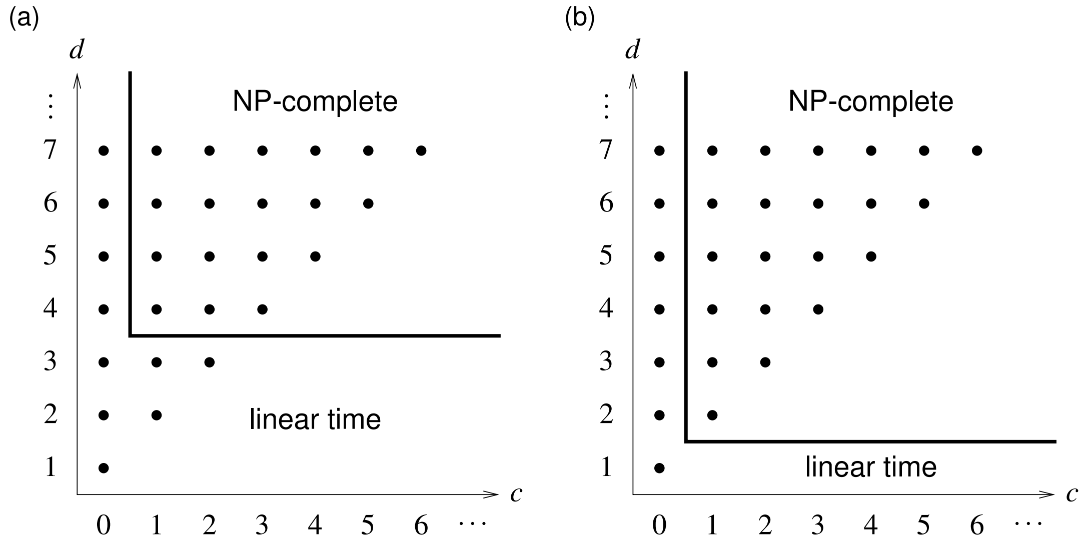

Figure 1: The dichotomies of the consistency and maximum consistency problems with coincidence c and sparseness d: **(a)** Consistency problems for the classes $\mathcal{B}$ and $\mathcal{B}^\ell$, where $\ell \geq 2$ (see Theorems 1 and 2); **(b)** maximum consistency problems for $\mathcal{B}, \mathcal{B}^\ell$, where $\ell \geq 1$, and $\mathcal{L}$ (see Theorems 3, 4, and 5). Linear time refers to the computing time required on a RAM model with uniform measure (see Section 2.4).

Finally, we present a dichotomy theorem for maximum consistency problems regarding neurons with arbitrary weights.

Theorem 5 MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d *is NP-complete whenever* $c \geq 1$ *and* $d \geq 2$. *On the other hand,* MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{L}$-CONSISTENCY WITH SPARSENESS d *for* $d \leq 1$ *is solvable in linear time.*

The dichotomy theorems are illustrated in Figure 1. The dot with coordinates (c,d) represents the consistency problem or the maximum consistency problem, respectively, with coincidence c and sparseness d. Note that for $c \geq d$ the problem at coordinate (c,d) can be identified with the problem at $(d-1,d)$ (see the discussion at the end of Section 2.3). Problems with sparseness 0 are omitted as they are trivial.

4. NP-Complete Problems

We establish the NP-completeness for the consistency problems of Theorems 1 and 2 in Section 4.1. Section 4.2 deals with the maximum consistency problems of Theorems 3, 4, and 5. In all cases it is sufficient to prove NP-hardness since membership in NP is easy to derive: Guessing the weights and the threshold and checking consistency or maximum consistency for the given function classes can be done in polynomial time.[3]

4.1 Consistency

The problem $\mathcal{B}$-CONSISTENCY without sample restrictions was shown to be NP-complete by Pitt and Valiant (1988). They have defined a reduction from the problem ZERO-ONE INTEGER PROGRAMMING that uses samples of unbounded coincidence and, hence, unbounded sparseness. To prove the stronger result, we make a new approach and establish a reduction from a problem that is a variant of the satisfiability problem which we call ALMOST DISJOINT POSITIVE 1-IN-3SAT.

ALMOST DISJOINT POSITIVE 1-IN-3SAT
Instance: A set U of variables, a collection $\mathcal{C}$ of subsets of U such that each subset $C \in \mathcal{C}$ has exactly three elements and each pair of subsets $C, D \in \mathcal{C}, C \neq D$, satisfies $|C \cap D| \leq 1$.
Question: Is there a truth assignment $\alpha : U \to \{0,1\}$ such that each subset in $\mathcal{C}$ has exactly one true variable?

The attribute "positive" refers to the fact that the variables are not negated; "almost disjoint" means that two subsets have at most one variable in common. The problem POSITIVE 1-IN-3SAT is known to be NP-complete (Garey and Johnson, 1979, p. 259). That the subproblem is NP-complete as well is claimed by the following statement. Its proof is given in the appendix.

Lemma 6 ALMOST DISJOINT POSITIVE 1-IN-3SAT *is NP-complete.*

This fact is employed in the following result. It covers the NP-completeness part of in Theorem 1. (Note that, as was argued in Section 2.3, NP-hardness of the consistency problem with coincidence c and sparseness d implies NP-hardness for any coincidence $c' \geq c$ and sparseness $d' \geq d$.)

Theorem 7 $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 4 *is NP-complete.*

Proof The proof is by reduction from the problem ALMOST DISJOINT POSITIVE 1-IN-3SAT defined above. Assume that an instance of this problem is given by the set of variables $U = \{u_1, \ldots, u_n\}$ and the collection of subsets $\mathcal{C} = \{c_1, \ldots, c_m\}$. The reduction maps this instance to a sample $S \subseteq \{0,1\}^{4n+m+2} \times \{0,1\}$. We introduce the following notation: Given a set $\{i_1, \ldots, i_j\} \subseteq \{1, \ldots, k\}$, let $[i_1, \ldots, i_j]_k$ denote that element of $\{0,1\}^k$ which has a 1 in positions $i_1, \ldots, i_j$, and 0 elsewhere. Then S is constructed as follows: For each variable $u_i \in U$, we define four examples

$$
\begin{aligned}
[i, n+i]_{4n+m+2} &\in \mathrm{Neg}(S), && (1)\\
[2n+i, 3n+i]_{4n+m+2} &\in \mathrm{Neg}(S), && (2)\\
[i, 3n+i, 4n+m+2]_{4n+m+2} &\in \mathrm{Pos}(S), && (3)\\
[n+i, 2n+i, 4n+m+2]_{4n+m+2} &\in \mathrm{Pos}(S). && (4)
\end{aligned}
$$

3. For the function class $\mathcal{L}$ one makes use of the fact that on binary inputs a linear threshold neuron with arbitrary weights needs no more than polynomially in n many bits to represent weights and threshold (see, e.g., Schmitt, 1994b).

Position i	Value of w_i
$1,\ldots,n$	$\alpha(u_i)$
$n+1,\ldots,2n$	$1-\alpha(u_i)$
$2n+1,\ldots,3n$	$\alpha(u_i)$
$3n+1,\ldots,4n$	$1-\alpha(u_i)$
$4n+1,\ldots,4n+m$	1
$4n+m+1$	1
$4n+m+2$	1

Table 1: Definition of the weights for a given truth assignment α.

Each subset $c_\ell = \{u_i, u_j, u_k\} \in \mathcal{C}$ gives rise to three examples

$$\begin{aligned} [i, j, k, 4n+\ell]_{4n+m+2} &\in \operatorname{Pos}(S), && (5)\\ [n+i, n+j, n+k]_{4n+m+2} &\in \operatorname{Pos}(S), && (6)\\ [4n+\ell, 4n+m+2]_{4n+m+2} &\in \operatorname{Pos}(S). && (7) \end{aligned}$$

Finally, we add the three examples

$$\begin{aligned} [4n+m+1]_{4n+m+2} &\in \operatorname{Neg}(S), && (8)\\ [4n+m+2]_{4n+m+2} &\in \operatorname{Neg}(S), && (9)\\ [4n+m+1, 4n+m+2]_{4n+m+2} &\in \operatorname{Pos}(S). && (10) \end{aligned}$$

Obviously, S can be computed from U and $\mathcal{C}$ in polynomial time. Further, as can easily be checked, we have $\operatorname{Coin}(S) \leq 1$ and $\operatorname{Sparse}(S) \leq 4$. (Note that every pair of different subsets in $\mathcal{C}$ has at most one common variable. Hence, the examples in (5) and (6) that arise from different subsets have coincidence at most 1.)

It remains to show that the reduction is correct, that is, that $\mathcal{C}$ has a truth assignment with exactly one true variable in each subset if and only if there exists a function in $\mathcal{B}_{4n+m+2}$ that is consistent with S. In order to prove the "only if" part, assume that $\alpha : U \to \{0,1\}$ is a truth assignment that satisfies the assumption. Define the weights $w_1, \ldots, w_{4n}$ as follows: For $i = 1, \ldots, n$, let

$$\begin{aligned} w_i &= w_{2n+i} = \alpha(u_i),\\ w_{n+i} &= w_{3n+i} = 1-\alpha(u_i). \end{aligned}$$

The remaining weights and the threshold are determined by

$$w_{4n+1} = \cdots = w_{4n+m+2} = 1 \quad \text{and} \quad t = 2.$$

The representation of the truth assignment by the weights is shown in Table 1. Using the fact that α assigns exactly one 1 to each subset in $\mathcal{C}$, it is easy to verify that the function represented by these parameter values is consistent with S.

For establishing the "if" part, let the weights $w_1, \ldots, w_{4n+m+2} \in \{0,1\}$ and the threshold t represent a function that is consistent with S. The examples in (8), (9), and (10) imply that

$$\begin{aligned} w_{4n+m+1} &< t,\\ w_{4n+m+2} &< t,\\ w_{4n+m+1} + w_{4n+m+2} &\geq t, \end{aligned}$$

from which it follows that $w_{4n+m+1} = 1$ and $w_{4n+m+2} = 1$. Hence, the threshold satisfies $1 < t \leq 2$, and we may assume without loss of generality that $t = 2$. Then the examples in (7) entail that $w_{4n+1} = \cdots = w_{4n+m} = 1$. From the examples in (1) to (4) and the facts that $w_{4n+m+2} = 1$ and $t = 2$, we derive the inequalities

$$\begin{aligned} w_i + w_{n+i} &\leq 1, \\ w_{2n+i} + w_{3n+i} &\leq 1, \\ w_i + w_{3n+i} &\geq 1, \\ w_{n+i} + w_{2n+i} &\geq 1, \end{aligned}$$

for $i = 1, \ldots, n$. These imply the equalities

$$w_i + w_{n+i} = 1, \tag{11}$$

for $i = 1, \ldots, n$. We define the truth assignment $\alpha : U \to \{0, 1\}$ by

$$\alpha(u_i) = w_i,$$

for $i = 1, \ldots, n$. The examples in (5), (6) and the fact that $t = 2$ yield the inequalities

$$\begin{aligned} w_i + w_j + w_k + w_{4n+\ell} &\geq 2, \\ w_{n+i} + w_{n+j} + w_{n+k} &\geq 2, \end{aligned}$$

for each subset $c_\ell = \{u_i, u_j, u_k\} \in \mathcal{C}$. From these, the fact that $w_{4n+1} = \cdots = w_{4n+m} = 1$, and the equations (11) we get that c_ℓ satisfies

$$\alpha(u_i) + \alpha(u_j) + \alpha(u_k) = 1.$$

Thus, each subset in $\mathcal{C}$ has exactly one 1 assigned by α. This completes the correctness proof for the reduction. ∎

The foregoing proof shows that the same reduction can be used whenever the threshold is allowed to take on the value 2. Thus, it follows that the problems in Theorem 7 are NP-complete also for neurons with threshold at most ℓ, where $\ell \geq 2$. This observation establishes the NP-completeness statement of Theorem 2.

Corollary 8 *For every $\ell \geq 2$, $\mathcal{B}^\ell$-*CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 4 *is NP-complete.*

4.2 Maximum Consistency

Now, we address the NP-completeness of the maximum consistency problems. The following result covers the intractability part of Theorem 3.

Theorem 9 MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

Proof Höffgen et al. (1995) have shown that the maximum consistency problem for the class $\mathcal{L}$ and for samples without restrictions is NP-complete. They constructed a reduction from VERTEX COVER (Garey and Johnson, 1979, p. 190). This is the problem to decide, given a graph $G = (V,E)$ and a number k, whether there exists a set $V' \subseteq V$ with $|V'| \leq k$ such that for each edge $\{u,v\} \in E$ at least one of u and v belongs to V'. We use this problem for the reduction, too, but we take account of the fact that the weights are from the set $\{0,1\}$.

Let (V,E) and k be an instance of VERTEX COVER, where $V = \{v_1,\ldots,v_n\}$. We define the sample $S \subseteq \{0,1\}^{2n} \times \{0,1\}$ using the notation introduced in the proof of Theorem 7. For each $v_i \in V$, there is a "vertex example"

$$[i,n+i]_{2n} \quad \in \quad \mathrm{Neg}(S).$$

Each $e = \{v_i,v_j\} \in E$ results in two "edge examples"

$$\begin{aligned} [i,j]_{2n} \quad &\in \quad \mathrm{Pos}(S), \\ [n+i,n+j]_{2n} \quad &\in \quad \mathrm{Pos}(S). \end{aligned}$$

The cardinality of the subsample in the instance of the maximum consistency problem is set to the value $|S|-k$. It is obvious that $\mathrm{Coin}(S) \leq 1$ and $\mathrm{Sparse}(S) \leq 2$ holds, and that the reduction is computable in polynomial time.

We claim that (V,E) has a vertex cover of cardinality at most k if and only if there exists a function in $\mathcal{B}_{2n}$ that is consistent with at least $|S|-k$ examples. Let $V' \subseteq V$ be a vertex cover with at most k elements. Define the weights $w_1,\ldots,w_{2n}$ by

$$w_i \quad = \quad w_{n+i} \quad = \quad \begin{cases} 1 & \text{if } v_i \in V', \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1,\ldots,n$, and let the threshold be $t = 1$. Then, the vertex example $[i,n+i]_{2n}$ is classified correctly if and only if $v_i \in V \setminus V'$. Further, since V' is a vertex cover, all edge examples are classified correctly. Thus, the function computed by this neuron is consistent with at least $|S|-k$ elements.

On the other hand, let the weights $w_1,\ldots,w_{2n}$ and the threshold t represent a function that is consistent with a subsample of S of cardinality at least $|S|-k$. Define a set V' of vertices as follows: For each vertex example that is classified wrongly include the corresponding vertex in V'; for each edge example that is classified wrongly arbitrarily choose one of the vertices the edge is incident with and include it in V'. Consequently, V' contains at most k elements. In order to show that V' is a vertex cover, assume that $\{v_i,v_j\} \in E$ is not covered, that is, we have $\{v_i,v_j\} \cap V' = \emptyset$. Then, by the construction of V', the function is consistent with the two vertex examples that arose from v_i and v_j, and it is consistent with the two edge examples due to $\{v_i,v_j\}$. Thus, we get

$$\begin{aligned} w_i + w_{n+i} + w_j + w_{n+j} \quad &< \quad 2t, \\ w_i + w_j + w_{n+i} + w_{n+j} \quad &\geq \quad 2t, \end{aligned}$$

a contradiction. This implies that V' is a vertex cover. ■

The reduction defined in the previous proof remains valid under the additional requirement that the threshold satisfies $t \leq 1$. As a consequence, we obtain the NP-completeness result claimed by Theorem 4.

Corollary 10 *For every $\ell \geq 1$,* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

Furthermore, we observe that the proof of Theorem 9 does not make use of the assumption that the weights must be from the set $\{0,1\}$. Consequently, the reduction works also for unconstrained weights, that is, for the function class $\mathcal{L}$. This establishes the NP-completeness statement of Theorem 5.

Corollary 11 MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

5. Problems Solvable in Linear Time

In the following, we show that all consistency problems and maximum consistency problems not yet considered in the foregoing section have algorithms that run in linear time on a RAM as made precise in Section 2.4. Thus, if P $\neq$ NP holds, the values of coincidence and sparseness established in the NP-completeness results are optimal. Moreover, as any algorithm that solves a consistency or maximum consistency problem must pass through the entire sample, the linear-time bound cannot be improved either. Thus, we not only obtain a complete characterization of the complexity of consistency problems for the function classes $\mathcal{B}$, $\mathcal{B}^\ell$, and $\mathcal{L}$ with sample restrictions, but also reveal the striking fact that, with respect to the RAM computer model, these classes consist solely of problems of the lowest and highest complexity in NP. All algorithms presented here are constructive, that is, they solve the decision problem by computing a solution if one exists.

5.1 Consistency

It is convenient to begin with the consistency problem for neurons with a bounded threshold. The case that remains to be considered is the function class $\mathcal{B}^1$. The result for this class implies the second part of Theorem 2. It is valid for samples with arbitrary coincidence and sparseness.

Theorem 12 $\mathcal{B}^1$-CONSISTENCY *is solvable in linear time.*

Proof If there are weights $w_1,\ldots,w_n \in \{0,1\}$ and a threshold $t \in \{0,1\}$ consistent with a given sample S then we may set $t=0$ if $\mathrm{Neg}(S)=\emptyset$. On the other hand, if S contains negative examples, we must have $t=1$. Then we ensure that $w \cdot x = 0$ for all $x \in \mathrm{Neg}(S)$. These are the steps performed by Algorithm 1, which is basically the standard consistency algorithm for disjunctions. The most time consuming part is the for-loop in lines 8–12. It is obvious that this statement can be implemented such that it requires no more than linear time on a RAM. ■

It is easy to see that if some function in $\mathcal{B}$ is consistent with a sample S that has coincidence 0, the threshold can be chosen to satisfy $t \leq 1$. Therefore, Algorithm 1 solves also the problem $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0.

Corollary 13 $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 *is solvable in linear time.*

Thus, the first of the two statements in Theorem 1 that claim a linear time bound is covered. The second statement is established by the following result.

Algorithm 1 $\mathcal{B}^1$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$

Output: $w_1, \ldots, w_n, t$

```
for i = 1,...,n do
    w_i := 1
end for;
if Neg(S) = ∅ then
    t := 0;
else
    t := 1;
    for all x ∈ Neg(S) do
        if there is some i ∈ {1,...,n} with x_i = 1 then
            w_i := 0
        end if
    end for
end if.
```

Theorem 14 $\mathcal{B}$-CONSISTENCY WITH SPARSENESS 3 *is solvable in linear time.*

Proof We show that Algorithm 2 computes a solution for samples with sparseness 3 provided that one exists. Without loss of generality we may assume that the threshold satisfies $t \leq 3$. The algorithm has three parts that correspond to the possible values for the threshold: $t \leq 1$, $t = 3$, and $t = 2$. In lines 1–2 Algorithm 1 is used to check whether there is a function in the class $\mathcal{B}^1$ consistent with S.

If this fails, the algorithm assumes in lines 3–12 that $t = 3$. Here, the assignment of values to the weights is based on the fact that every positive example must reach the threshold. Hence, its non-zero components, of which there must be exactly three, specify which weights receive the value 1.

In the last part, starting from line 13, we can only have that $t = 2$. We show that this case of the consistency problem can be solved via a 2-SATISFIABILITY problem. The algorithm maps the sample S to a set $\mathcal{C}$ of clauses over the variables $w_1, \ldots, w_n$. Each clause has at most two literals of the form w_i or $\neg w_i$. For each element $x \in \mathrm{Dom}(S)$ the set $\mathcal{C}$ is augmented with clauses that depend on the properties of x. Lines 17–23 deal with the case that x has a 1 in exactly three positions. If x is a positive example then no two weights at these positions may both be equal to 0. This is expressed by the three clauses in line 20. If x is a negative example then no two of these weights may both be equal to 1. This is taken into account by adding the three clauses in line 22. Lines 24–30 treat the examples with exactly two 1s. If x is positive then both weights must be equal to 1 (line 27); if x is negative then at least one weight must be 0 (line 29). Finally, if x has at most one non-zero position and x is positive then the consistency problem has no solution. This is taken care of in line 34 by adding two contradictory clauses. On the other hand, if x is negative then it will be classified as such and nothing needs to be done. In the last step an algorithm for 2-SATISFIABILITY is called with the constructed set of clauses as input.

It remains to verify that the algorithm runs in linear time on a RAM. Using Theorem 12 we infer that the part in lines 1–14 requires no more than linear time. The construction of the set of clauses in lines 15–37 is performed in linear time. (Note that each example gives rise to at most three clauses.)

Algorithms that compute satisfying assignments for 2-SATISFIABILITY problems in linear time on a RAM with uniform measure have been designed by Even et al. (1976) and Aspvall et al. (1979). It is obvious that the size of the constructed instance for 2-SATISFIABILITY is at most linear in the size of the sample. Thus, line 38 can be executed in linear time. We conclude that the running time of Algorithm 2 on a random access machine is linear. ■

With the previous result the proof of Theorem 1 is completed. Obviously, Algorithm 2 solves also the consistency problems with sparseness 3 for the classes $\mathcal{B}^\ell$ where $\ell \geq 2$. (If $\ell = 2$, the middle part of the algorithm, which assumes $t = 3$, is needless.) Furthermore, if some function in $\mathcal{B}^\ell$ is consistent with a sample having coincidence 0 then there is also some function in $\mathcal{B}^1$ consistent with that sample. Consequently, Algorithm 1 solves also the problems $\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 0 for every $\ell \geq 2$. Hence, Theorems and 14 and 12 yield the following statement. It finalizes the proof of Theorem 2.

Corollary 15 *For every $\ell \geq 2$, $\mathcal{B}^\ell$-*CONSISTENCY WITH COINCIDENCE 0 *and $\mathcal{B}^\ell$-*CONSISTENCY WITH SPARSENESS 3 *is solvable in linear time.*

5.2 Maximum Consistency

At last, we give attention to the maximum consistency problems that are claimed as linear-time solvable in Theorems 3, 4, and 5. The main issues that have to be dealt with are contradictory pairs and examples that are 0 in all positions. With the first result we finish the proof of Theorem 3.

Theorem 16 *The problems* MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{B}$-CONSISTENCY WITH SPARSENESS 1 *are solvable in linear time.*

Proof If $\mathrm{Neg}(S) = \emptyset$ then by letting $w_1 = \cdots = w_n = 0$ and $t = 0$ we obtain the function that always outputs 1, which is consistent with all examples. Assume now that $\mathrm{Neg}(S) \neq \emptyset$. Algorithm 3 generates a function that has the following properties: For each example $x \in \mathrm{Dom}(S)$ that is part of a contradictory pair, that is, where $(x,0) \in S$ and $(x,1) \in S$, the function is consistent with the positive example. Further, it is consistent with all examples that are not part of a contradictory pair, except possibly for the example $(0,\ldots,0) \in \mathrm{Pos}(S) \setminus \mathrm{Neg}(S)$, which is classified as negative. It is obvious that this is the maximum number of examples that can be classified correctly, unless $(0,\ldots,0) \in \mathrm{Pos}(S) \setminus \mathrm{Neg}(S)$ is the only example that is not part of a contradictory pair. In this case we can increase the number of correctly classified examples by using the function that constantly outputs 1, which is then consistent with the positive example in each contradictory pair and with $(0,\ldots,0) \in \mathrm{Pos}(S)$.

It is obvious that Algorithm 3 and the additional steps described above require no more than linear time on a RAM. In particular, whether $(0,\ldots,0)$ is the only example that is not part of a contradictory pair can be checked in linear time using radix sort.

If $\mathrm{Sparse}(S) = 1$ then we proceed in the same way. In order to see that this is correct, observe that $\mathrm{Coin}(S) = 1$ holds only if S contains examples $(x,0)$ and $(x,1)$. Thus, a maximum subsample S' that does not contain contradictory pairs satisfies $\mathrm{Coin}(S') = 0$. ■

Algorithm 2 $\mathcal{B}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1,\ldots,w_n,t$

get $w_1,\ldots,w_n,t$ from $\mathcal{B}^1$-CONSISTENCY(S);
if $(w_1,\ldots,w_n,t)$ is consistent with S **then stop end if**;
$t := 3$;
for $i = 1,\ldots,n$ **do**
  $w_i := 0$
end for;
for all $x \in \mathrm{Pos}(S)$ **do**
  if there is some $x \in \mathrm{Pos}(S)$ with $x_i = 1$ **then**
    $w_i := 1$
  end if
end for;
if $(w_1,\ldots,w_n,t)$ is consistent with S **then stop end if**;
$t := 2$;
$\mathcal{C} := \emptyset$;
for all $x \in \mathrm{Dom}(S)$ **do** {construct a set $\mathcal{C}$ of clauses over $w_1,\ldots,w_n$}
  let $I = \{i \mid x_i = 1\}$;
  if $|I| = 3$ **then**
    let $\{j,k,\ell\} = I$;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j,w_k\},\{w_j,w_\ell\},\{w_k,w_\ell\}\big\}$
    else {we have $x \in \mathrm{Neg}(S)$}
      $\mathcal{C} := \mathcal{C} \cup \big\{\{\neg w_j,\neg w_k\},\{\neg w_j,\neg w_\ell\},\{\neg w_k,\neg w_\ell\}\big\}$
    end if
  else if $|I| = 2$ **then**
    let $\{j,k\} = I$;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j\},\{w_k\}\big\}$
    else {we have $x \in \mathrm{Neg}(S)$}
      $\mathcal{C} := \mathcal{C} \cup \big\{\{\neg w_j,\neg w_k\}\big\}$
    end if
  else {we have $|I| \leq 1$}
    if $|I| = 1$ **then** let $\{j\} = I$ **else** $j := 1$ **end if**;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j\},\{\neg w_j\}\big\}$
    end if
  end if
end for;
get $w_1,\ldots,w_n$ from 2-SATISFIABILITY($\{w_1,\ldots,w_n\},\mathcal{C}$).

The algorithm presented in the foregoing proof generates only solutions where the threshold is at most 1. Thus, it solves also the corresponding problems for neurons with a bounded threshold as was claimed in the first statement of Theorem 4.

Algorithm 3 MAX-$\mathcal{B}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1, \ldots, w_n, t$

```
t := 1;
for i = 1,...,n do
    w_i := 0
end for;
for all x ∈ Pos(S) do
    if there is some i ∈ {1,...,n} with x_i = 1 then
        w_i := 1
    end if
end for.
```

Corollary 17 *For every $\ell \geq 1$,* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH SPARSENESS 1 *is solvable in linear time.*

The following result completes the proof of Theorem 4.

Lemma 18 MAX-$\mathcal{B}^0$-CONSISTENCY *is solvable in linear time.*

Proof If the threshold is equal to 0 then everything is classified as positive regardless of which values the weights may have. Thus, outputting any weights and $t = 0$ provides a solution if one exists. For solving the decision problem, observe that consistency with at least k examples in S is equivalent to the condition $|\text{Pos}(S)| \geq k$. This can clearly be checked in linear time on a RAM. ■

Finally, we prove the remaining part of Theorem 5 that claims linear time bounds.

Theorem 19 *The problems* MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{L}$-CONSISTENCY WITH SPARSENESS 1 *are solvable in linear time.*

Proof Consider Algorithm 4 for inputs satisfying $\text{Coin}(S) = 0$. If $\text{Neg}(S) = \emptyset$ then the resulting function is consistent with all examples. In the case that $\text{Neg}(S) \neq \emptyset$ holds we argue as in the proof of Theorem 16: For every contradictory pair, the function is consistent with the positive example. Further, it is consistent with every example that is not part of a contradictory pair. (In contrast to the proof of Theorem 16, this holds also in all cases where $(0, \ldots, 0) \in \text{Dom}(S)$ due to the use of negative weights by Algorithm 4.) Clearly, this is the maximum number of examples a function can be consistent with.

Since any subsample S' of S that contains no contradictory pairs must satisfy $\text{Coin}(S') = 0$ if $\text{Sparse}(S) = 1$, Algorithm 4 solves also the maximum consistency problem for samples with sparseness 1.

That the running time of Algorithm 4 is linear on a RAM is obvious. ■

Algorithm 4 MAX-$\mathcal{L}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1,\ldots,w_n,t$

```
if (0,...,0) ∈ Pos(S) then
    t := 0
else
    t := 1
end if;
for i = 1,...,n do
    w_i := −1
end for;
for all x ∈ Pos(S) do
    if there is some i ∈ {1,...,n} with x_i = 1 then
        w_i := 1
    end if
end for.
```

6. Conclusions

With this work we aimed at providing necessary and sufficient conditions for the existence of efficient learning algorithms. The approach that we have taken is to consider criteria for the complexity of samples and to use them for the definition of subproblems. As results we obtained dichotomy theorems that completely characterize the dividing lines between the polynomial-time solvable and the NP-complete problems.

An NP-completeness assertion is a worst-case result stating that, if P $\neq$ NP then there is no algorithm that solves the problem in polynomial time for all instances. One way to cope with intractability is therefore trying to avoid instances that make the problem hard. This gives rise to subproblems that may be more appropriate to practical situations where one often needs not deal with all possible instances but only a subset thereof. The results here show that such deliberations can indeed lead to the discovery of new efficient learning algorithms. While it is not clear whether the restrictions they suppose are always met in applications, these algorithms could be helpful in the development of better heuristics for more general learning problems.

A number of questions arises from this work that are worth further investigation. We have considered only two types of neural "networks": single neurons with binary and with arbitrary weights. Whether dichotomy theorems can be established for other neuron types and for neural networks is an interesting open problem. Further, the problems we have studied can be generalized by allowing examples with integer or even rational components. While the NP-completeness results remain valid in these cases, it is not obvious how the algorithms can be adapted to deal with non-binary inputs. Finally, we have introduced here two specific criteria for sample restrictions. Motivated by theoretical or practical considerations one can think of many other ways to select such conditions.

Acknowledgments

I thank the anonymous reviewers for their helpful comments and suggestions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. An NP-Complete Satisfiability Problem

Lemma 6 in Section 4.1 claims that the problem ALMOST DISJOINT POSITIVE 1-IN-3SAT introduced in that section is NP-complete. Here we give a proof of this statement.

Lemma 6 ALMOST DISJOINT POSITIVE 1-IN-3SAT *is NP-complete.*

Proof It is clear that the problem is in NP. To prove its NP-hardness, we construct a reduction from POSITIVE 1-IN-3SAT. In this variant of the problem the requirement is missing that two subsets have at most one variable in common. The idea of the reduction is to establish the property of almost disjointness by introducing new variables as substitutes for old ones. The correctness of the reduction is ensured by augmenting the collection with additional subsets.

Let $\mathcal{C}$ be the collection of subsets from the instance of POSITIVE 1-IN-3SAT. Further, let $C,D \in \mathcal{C}$ be two subsets with two common variables, say $C = \{u_1,u_2,u_3\}$ and $D = \{u_1,u_2,u_4\}$. We introduce a set of new variables $V = \{v_1,\ldots,v_5\}$, replace in D the variable u_1 by v_1 and join $\mathcal{C}$ with the collection $\mathcal{D}$ defined as

$$\mathcal{D} \quad = \quad \big\{\{v_1,v_2,v_3\},\ \{v_1,v_4,v_5\},\ \{u_1,v_2,v_4\},\ \{u_1,v_3,v_5\}\big\}.$$

Obviously, no pair of subsets in $\mathcal{D}$ has more than one common variable. Further, no subset in $\mathcal{D}$ has more than one variable in common with any subset in $\mathcal{C}$. Hence, going from $\mathcal{C}$ to $\mathcal{C} \cup \mathcal{D}$, the number of pairs that violate the condition of almost disjointness decreases by one.

We say that a truth assignment is a 1-IN-3SAT assignment for a collection of subsets if each subset has exactly one true variable. Let U be the set of variables in $\mathcal{C}$. We claim that the following two conditions hold:

(A.1) Every 1-IN-3SAT assignment $\alpha : U \to \{0,1\}$ for $\mathcal{C}$ can be extended to a 1-IN-3SAT assignment $\alpha : U \cup V \to \{0,1\}$ for $\mathcal{C} \cup \mathcal{D}$ satisfying $\alpha(u_1) = \alpha(v_1)$.

(A.2) Every 1-IN-3SAT assignment $\alpha : U \cup V \to \{0,1\}$ for $\mathcal{C} \cup \mathcal{D}$ satisfies $\alpha(u_1) = \alpha(v_1)$.

That condition (A.1) is valid can be seen as follows: Given α defined on U, we let $\alpha(v_1) = \alpha(u_1)$ and extend it to the remaining variables of V in the following way: In the case $\alpha(u_1) = 0$ we define $\alpha(v_2) = \alpha(v_5) = 0$ and $\alpha(v_3) = \alpha(v_4) = 1$; in the case $\alpha(u_1) = 1$ we let $\alpha(v_2) = \alpha(v_3) = \alpha(v_4) = \alpha(v_5) = 0$. For the proof of condition (A.2) we observe that if $\alpha(v_1) = 1$, we must have $\alpha(v_2) = \alpha(v_3) = \alpha(v_4) = \alpha(v_5) = 0$ due to the first two subsets in $\mathcal{D}$, and hence, because of the last two subsets, $\alpha(u_1) = 1$; on the other hand, if $\alpha(u_1) = 1$, analogous reasoning yields that $\alpha(v_1) = 1$ must hold. Thus, u_1 and v_1 cannot have different truth values assigned by α.

Conditions (A.1) and (A.2) imply that $\mathcal{C}$ has a 1-IN-3SAT assignment if and only if $\mathcal{C} \cup \mathcal{D}$ has a 1-IN-3SAT assignment. Repeating the above described procedure for every pair of subsets with two common variables we finally arrive at a collection that is almost disjoint. The correctness of the reduction follows inductively. It is obvious that the reduction is computable in polynomial time. Thus, ALMOST DISJOINT POSITIVE 1-IN-3SAT is NP-hard. ■

References

Amaldi, E. (1991). On the complexity of training Perceptrons. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 55–60. Elsevier, Amsterdam.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

Aspvall, B., Plass, M. F., and Tarjan, R. E. (1979). A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8:121–123.

Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36:929–965.

Cook, S. A. and Reckhow, R. A. (1973). Time bounded random access machines. *Journal of Computer and System Sciences*, 7:354–375.

Dalmau, V. (1999). A dichotomy theorem for learning quantified Boolean formulas. *Machine Learning*, 35:207–224.

Dalmau, V. and Jeavons, P. (2003). Learnability of quantified formulas. *Theoretical Computer Science*, 306:485–511.

Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703.

Fang, S. C. and Venkatesh, S. S. (1996). Learning binary Perceptrons perfectly efficiently. *Journal of Computer and System Sciences*, 52:374–389.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco.

Golea, M. and Marchand, M. (1993). On learning Perceptrons with binary weights. *Neural Computation*, 5:767–782.

Hampson, S. E. and Volper, D. J. (1986). Linear function neurons: Structure and training. *Biological Cybernetics*, 53:203–217.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, second edition.

Höffgen, K.-U., Simon, H.-U., and Van Horn, K. S. (1995). Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50:114–125.

Judd, J. S. (1990). *Neural Network Design and the Complexity of Learning*. The MIT Press, Cambridge, MA.

Kearns, M. J., Schapire, R. E., and Sellie, L. M. (1994). Toward efficient agnostic learning. *Machine Learning*, 17:117–141.

Kim, J. H. and Roche, J. R. (1998). Covering cubes by random half cubes, with applications to binary neural networks. *Journal of Computer and System Sciences*, 56:223–252.

Kirousis, L. M. and Kolaitis, P. G. (2003). The complexity of minimal satisfiability problems. *Information and Computation*, 187:20–39.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

Maass, W. and Turán, G. (1994). How fast can a threshold gate learn? In Hanson, S. J., Drastal, G., and Rivest, R., editors, *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, pages 381–414. The MIT Press, Cambridge, MA.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Natarajan, B. K. (1991). *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, San Mateo, CA.

Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36:19–31.

Pitt, L. and Valiant, L. G. (1988). Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35:965–984.

Schaefer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226.

Schmitt, M. (1994a). On the complexity of consistency problems for neurons with binary weights. Ulmer Informatik-Berichte 94-01, Universität Ulm.

Schmitt, M. (1994b). On the size of weights for McCulloch-Pitts neurons. In Caianiello, E. R., editor, *Proceedings of the Sixth Italian Workshop on Neural Nets WIRN VIETRI-93*, pages 241–246, World Scientific, Singapore.

Schmitt, M. (1995). On methods to keep learning away from intractability. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of the International Conference on Artificial Neural Networks ICANN '95*, volume 1, pages 211–216, EC2 & Cie., Paris.

Servedio, R. A. (2002). Perceptron, Winnow, and PAC learning. *SIAM Journal on Computing*, 31:1358–1369.

Sommer, F. T. and Palm, G. (1999). Improved bidirectional retrieval of sparse patterns stored by Hebbian learning. *Neural Networks*, 12:281–297.

van Emde Boas, P. (1990). Machine models and simulations. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 1, pages 1–66. Elsevier, Amsterdam.

Šíma, J. (2002). Training a single sigmoidal neuron is hard. *Neural Computation*, 14:2709–2728.

Zheng, L. and Stuckey, P. J. (2002). Improving SAT using 2SAT. In Oudshoorn, M. J., editor, *Proceedings of the Twenty-Fifth Australasian Computer Science Conference ACSC2002*, volume 4 of *Conferences in Research and Practice in Information Technology*, pages 331–340, Australian Computer Society, Sydney.