From Finding Maximum Feasible Subsystems of Linear Systems to Feedforward Neural Network Design

Présentée au Département de Mathématiques

École Polytechnique Fédérale de Lausanne

Pour l'obtention du grade de docteur ès sciences

Par

Edoardo Amaldi
Ingénieur mathématicien diplômé
de l'École Polytechnique Fédérale de Lausanne
de nationalités italienne et suisse

acceptée sur proposition du jury :

Prof. Dominique de Werra, rapporteur
Prof. Yaser Abu-Mostafa, corapporteur
Prof. Martin Hasler, corapporteur
Prof. Jean-Philippe Vial, corapporteur

Lausanne, EPFL
1994

To the memory of my grandparents
Edoardo and Ginestra Amaldi

Abstract

The central theme of this dissertation is the important class of problems that consist, given a possibly infeasible linear system, of finding a solution that satisfies a maximum number of relations or violates a minimum number of them. Many variants of these problems are attracting considerable attention in a wide range of fields such as operations research, pattern recognition and machine learning. Previous and ongoing works include a number of exact and heuristic algorithms for tackling various special cases. Before we started our work, not much was known on the intrinsic complexity of solving this widespread class of problems. What is more, their approximability was unresolved even though approximate solutions are of great practical value.

In the first part of this thesis we consider the main variants of these combinatorial problems within a unified framework and investigate their inherent complexity, independently from any particular algorithm. Building on the recent advances in computational complexity, we study in detail the two complementary problems with any of the four types of relational operators $=$, $\geq$, $>$ and $\neq$. Besides the basic versions, particular attention is devoted to important variants with mixed types of relational operators, to weighted versions where a different importance can be assigned to each relation and to constrained ones with mandatory relations or discrete variables. Since in practice approximate solutions often suffice, we determine the complexity of solving these problems to optimality and, whenever they are intractable, we try to determine their degree of approximability as precisely as possible. Our collection of results shows that the approximability of similar variants can differ enormously and it allows to locate them within the emerging hierarchy of approximation classes.

The second part of this thesis is devoted to the special variants of these combinatorial problems arising in the rapidly growing field of artificial neural networks. The focus is on feedforward networks of linear threshold units.

First we discuss the implications of our complexity results on the difficulty of training this type of networks. In particular, we disprove the "folk" theorem stating that training single-layer networks is easy while it becomes intractable as soon as there are hidden units. However, we show that the situation is not as bad as it has subsequently been claimed by some other authors. Based on this discussion which goes beyond artificial neural networks, we propose an extension of the famous PAC model of learning from examples that makes

it more realistic. The fundamental issue of designing close-to-minimum size networks is also addressed with a special focus on nonzero weights instead of computing units. In particular, we prove that polynomial time algorithms guaranteed to design near-optimum perceptrons with a number of nonzero weights that is at most logarithmically larger than the minimum one are extremely unlikely.

The intrinsic complexity of training even single-layer networks and designing compact perceptrons, compels us to look for efficient heuristics. Our interest is in simple methods that require local information and are reasonable candidates for hardware implementation.

Two variants of the famous perceptron algorithm are presented for minimizing the number of errors made by a single unit. The first one is a version of the thermal rule that makes it adaptive and more efficient. The second one is a probabilistic variant that requires lower precision computations and has an asymptotic convergence guarantee. The numerical experiments carried out on simulated and real-world data show that our variants converge much faster than the competing ones while providing comparable solutions.

Three constructive methods for designing multilayer networks are then presented. Starting with a single-layer network, new units are allocated incrementally and each of them is trained using the above-mentioned procedures. Unlike in conventional training algorithms, in constructive methods the topology does not need to be guessed a priori and the computational burden is much lower. A set of experiments indicates that our SHIFT method compares favorably with various competing strategies for building multilayer networks. Since constructive algorithms are based on a greedy approach, it turns out that it is not necessary and even desirable to train each unit optimally. In fact, if efficient variants of the perceptron procedure are used to train single units, our simple constructive strategy produces rapidly compact networks with good generalization properties.

Résumé

Le thème central de cette thèse est l'étude d'une classe importante de problèmes qui consistent, étant donné un système linéaire, à déterminer une solution satisfaisant un nombre maximum de relations ou en violant un nombre minimum. De nombreuses variantes de ces problèmes se rencontrent dans des domaines aussi variés que la recherche opérationnelle, la reconnaissance de formes et l'apprentissage automatique. Les travaux de recherche achevés et en cours portent principalement sur des méthodes exactes et heuristiques visant plusieurs cas particuliers. Avant ce travail, on ne savait que peu de chose de la complexité de cette vaste classe de problèmes et, bien que les solutions approchées soient d'une importance capitale en pratique, leur niveau d'approximabilité n'avait pas encore été étudié.

Dans la première partie de cette thèse nous considérons les principales variantes de ces problèmes combinatoires dans un même cadre et nous étudions leur degré d'approximabilité intrinsèque, indépendamment de tout algorithme. Sur la base des récents développements en théorie de la complexité, nous analysons en détail les deux problèmes complémentaires en envisageant les quatre principaux opérateurs $=$, $\geq$, $>$ et $\neq$. Partant des versions de base, nous considérons également les variantes contenant divers types d'opérateurs, les versions pondérées dans lesquelles un poids propre peut être attribué à chaque relation et les versions contraintes contenant des relations obligatoires ou des variables astreintes à prendre des valeurs discrètes. Les solutions approchées étant souvent satisfaisantes, nous établissons la complexité de ces divers problèmes en termes de solutions optimales et, lorsqu'ils ne peuvent être résolus en temps polynomial, nous déterminons aussi précisément que possible leur degré d'approximabilité. Nos résultats montrent que des variantes apparemment semblables peuvent avoir des propriétés d'approximation très differentes. De plus, ils nous permettent de les situer à l'intérieur de la hiérarchie des classes de complexité qui est en cours de d'élaboration.

La deuxième partie de cette thèse est consacrée aux cas particuliers de ces problèmes que l'on rencontre dans le domaine, en plein essor, des réseaux de neurones artificiels. Dans ce cadre, nous nous intéressons aux réseaux multicouches à unités binaires à seuil.

Nous étudions tout d'abord les conséquences de nos résultats de complexité au niveau de la difficulté de l'apprentissage dans ce type de réseaux. En particulier, nous réfutons la conjecture communément admise selon laquelle le problème d'apprentissage dans les réseaux monocouches est facile et devient difficile dès que le réseau possède des unités cachées.

Cependant, nous montrons que ce problème n'est pas aussi difficile que d'autres auteurs l'on prétendu à la suite de nos premiers travaux. En plaçant ce résultat dans un contexte plus général, nous proposons une extension du célèbre modèle PAC d'apprentissage sur la base d'exemples qui le rend plus réaliste. Le problème fondamental de configuration de réseaux de taille minimum est également abordé en mettant l'accent sur le nombre de poids non nuls au lieu du nombre d'unités. Nous démontrons notamment qu'il n'existe très vraisemblablement pas d'algorithme fournissant des perceptrons dont le nombre de poids non nuls dépasse le minimum d'au plus un facteur logarithmique.

La complexité des problèmes d'apprentissage et de configuration même pour de simples perceptrons, ne permet d'envisager que des méthodes heuristiques. Dans ce cadre, nous nous intéressons à des algorithmes qui nécessitent des informations locales et qui pourraient faire l'objet de réalisations matérielles.

Deux variantes de la célèbre méthode du perceptron sont proposées afin de minimiser le nombre d'erreurs commises par une unité isolée. La première est une version de l'approche "thermique" qui ne nécessite pas de réglage explicite des paramètres et qui s'avère plus efficace. La deuxième est une variante probabiliste qui présente des avantages quant au calcul et qui possède une garantie de convergence asymptotique. Les résultats numériques montrent que nos variantes convergent plus rapidement que celles concurrentes, tout en fournissant des solutions d'aussi bonne qualité.

Enfin, nous proposons trois méthodes constructives pour la configuration de réseaux multicouches. Partant d'un réseau sans unités cachées, de nouvelles unités sont ajoutées de manière incrémentale et l'apprentissage de chacune d'entre elles est réalisé avec les procedures sus-mentionnées. Contrairement aux méthodes conventionnelles, les algorithmes constructifs ne nécessitent pas le choix de la topologie du réseau a priori et ils sont nettement moins coûteux en temps de calcul. Des expériences portant sur des données simulées et réelles indiquent que notre algorithme du SHIFT présente plusieurs avantages par rapport à plusieurs méthodes de même catégorie. Les approches constructives étant basées sur un principe "glouton", il ne s'avère pas nécessaire, ni même souhaitable, de réaliser un apprentissage optimal au niveau de chaque unité. En utilisant les variantes efficaces de la méthode du perceptron, notre stratégie constructive fournit des réseaux de petite taille qui possèdent de bonnes propriétés de généralisation.

Acknowledgments

I am very grateful to my thesis advisor, Professor Dominique de Werra, for attracting me with his competence and enthusiasm into the field of combinatorial optimization. His support and the freedom he left me during these years have been very stimulating factors.

I express my gratitude to Professors Yaser Abu-Mostafa, Sristhi Chatterji, Martin Hasler and Jean-Philippe Vial for serving on my thesis committee. I highly appreciated the complementary points of view with which they evaluated this work.

My interest in complexity theory and in the theoretical aspects of artificial neural networks was fostered by the exciting courses given by Yaser Abu-Mostafa and Stephen Judd during the year I spent in the Computation and Neural Systems Program, California Institute of Technology.

The collaboration with Claude Diderich, Bertrand Guenin and Viggo Kann on different parts of this work has been a pleasure and I thank each of them for this enriching experience. Through the long distance e-mail exchanges with Viggo I discovered the remarkable possibilities of modern networking in scientific collaboration.

Thanks also to my past and present colleagues of the Operations Research group for the pleasant and lively atmosphere. I am particularly grateful to Frédéric Aviolat and Eddy Mayoraz for intensive scientific exchanges and for careful reading of the manuscript.

The MANTRA Centre for Neuromimetic Systems has been a truly interdisciplinary forum where we could launch and develop some challenging applied projects. Thanks to all the members of the Centre for the many friendly and stimulating interactions.

I appreciated Ugo Amaldi's, Andrea Ferrero's, Giorgio Gallo's and Isabelle de La Moussaye's help in identifying language mistakes and incoherences in the manuscript.

Finally, I am very grateful to my parents for having taught me, by their example, to do whatever one undertakes with enthusiasm.

Contents

Chapter 1

Introduction

1.1 Maximum feasible subsystems of linear systems

In many scientific disciplines and fields of application, a variety of physical or mathematical problems can be modeled in terms of systems of linear relations. The wide range of applicability of such linear formulations is due to their relative simplicity as well as to the efficiency of the methods that have been developed to solve particular cases. Linearity plays a fundamental role both from a theoretical and practical point of view. While allowing to model many different situations, it requires that the problems being considered be casted into a relatively simple form.

It is well known that feasible systems of linear equalities or inequalities can be solved efficiently, i.e. in an amount of time which grows polynomially with the size of the instance. Gaussian elimination is probably the most widely used procedure for finding solutions of sets of simultaneous equations [44]. Early methods for solving systems of inequalities include Fourier-Motzkin elimination [44] and Agmon-Motzkin-Schoenberg relaxation procedure where only a single inequality is considered at each step [3, 133]. These methods are known to terminate in a finite (and in the worst-case exponential) number of steps. However, more recent Linear Programming algorithms, like Khachian's adaptation of the ellipsoid method [106] or Karmarkar's interior point technique [101], are guaranteed to solve systems of equalities or inequalities in polynomial time.

For infeasible systems, the best we can ask for is an optimal solution which satisfies the given relations as well as possible. What is meant by an optimal solution depends on the particular application and various objectives may be of interest. In several fields – such as linear regression– the focus is on solutions which minimize the least mean squared error over all relations. This approach has been extensively investigated, the concept of pseudo-inverse of a matrix has been introduced, and efficient methods for computing such solutions have been developed [34]. However, in many cases this quadratic criterion is not appropriate as it depends on all the satisfied as well as all the violated relations.

The central theme of this dissertation is an important class of problems related to finding approximate solutions of infeasible linear systems with any of the four basic types of relational operators $=$, $\geq$, $>$ and $\neq$. When the main interest is in feasible subsystems, two natural and closely related objectives arise. Given a possibly infeasible linear system, the goal may be to find a solution which simultaneously satisfies as many relations as possible. This general problem, which amounts to determining a maximum Feasible Linear Subsystem, will be referred to as MAX FLS. Instead of counting the satisfied relations, one may alternatively focus on the unsatisfied ones. This gives rise to the MIN ULR problem where the number of Unsatisfied Linear Relations has to be minimized. In other words, one looks for a smallest set of relations that must be removed from the given system to make it feasible.

Since MAX FLS and MIN ULR with the same type of relations only differ in the objective function, these complementary problems are equivalent as far as optimal solutions are concerned. Indeed, if p denotes the total number of relations, the optimal solutions of MAX FLS satisfying s relations correspond to the optimal solutions of MIN ULR violating $p - s$ relations. However, as we shall see, such pairs of problems turn out to be quite different when near-optimal solutions are considered.

Unlike minimizing the least mean squared criterion, optimizing the number of satisfied or unsatisfied relations gives rise to combinatorial problems with noncontinuous objective functions. In principle, these problems can be solved by enumerating all possible subsystems (finitely many) and checking whether they are feasible. But in practice such a brute force technique is hopeless even for instances of very moderate size and more efficient methods are required.

1.2 The many fields of occurrence

MAX FLS and MIN ULR are not only interesting as natural combinatorial problems from linear algebra but also because many variants occur in a number of fields such as operations research [94, 77], pattern recognition [163, 54] and machine learning [65, 6]. In this section we mention a few examples while keeping for the end the one related to artificial neural networks which is the focus of the second part of this dissertation. The list is not intended to be exhaustive but to give an idea of the importance and the variety of areas where this type of optimization problems arise. As we shall see, a number of variants with additional features or constraints have already attracted a considerable attention in the literature.

Before turning to the main fields of occurrence, we need a formal definition of the two classes of problems.

> MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A and an n-dimensional vector $\mathbf{b}$ both with rational coefficients, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which satisfies as many relations as possible.

> MIN ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A and an n-dimensional vector $\mathbf{b}$ both with rational coefficients, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which violates as few relations as possible while satisfying all the others.

The kth relation of $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ is denoted by $\mathbf{a}^k\mathbf{x}\,\mathcal{R}\,b_k$, where $\mathbf{a}^k$ with $1 \leq k \leq p$ is the kth row of the matrix A and b_k the kth component of $\mathbf{b}$. Clearly, when homogeneous systems with equalities or nonstrict inequalities are considered, the trivial solution $\mathbf{x} = \mathbf{0}$ is of no interest.

1.2.1 Computational geometry

Many MAX FLS and MIN ULR variants turn out to have interesting geometrical interpretations.

A first example pertains to MAX FLS$^{>}$ and MIN ULR$^{>}$. According to Carver's transposition theorem [151], a system $A\mathbf{x} < \mathbf{b}$ is feasible if and only if $\mathbf{y} = \mathbf{0}$ is the unique solution of

$$\begin{aligned} A^t\mathbf{y} &= \mathbf{0} \\ \mathbf{b}^t\mathbf{y} &\leq 0 \\ \mathbf{y} &\geq \mathbf{0}. \end{aligned} \tag{1.1}$$

Thus homogeneous MAX FLS$^{>}$ with $\mathbf{b} = \mathbf{0}$ is equivalent to finding among the p vectors $\{\mathbf{a}^k\}_{1 \leq k \leq p}$ a largest subset of *positive linearly independent* vectors, i.e. a maximum subset $\{\mathbf{a}^{k_j}\}_{1 \leq j \leq p'}$ such that $\sum_{j=1}^{p'} \alpha_{k_j}\mathbf{a}^{k_j} = \mathbf{0}$ with nonnegative coefficients $\alpha_{k_j} \geq 0$ necessarily implies that $\alpha_{k_j} = 0$ for all $1 \leq j \leq p'$. In terms of cones, one looks for a pointed cone that is generated by as many vectors in $\{\mathbf{a}^k\}_{1 \leq k \leq p}$ as possible [151]. Pointed cones correspond to the intuitive concept of a cone that has an extreme point. More precisely, a cone is *pointed* if and only if it contains no vector $\mathbf{x} \neq \mathbf{0}$ such that $-\mathbf{x}$ is also in the cone. These well-known geometrical objects are often of interest [151].

The basic versions of MAX FLS and MIN ULR with any type of relational operator have two simple geometrical interpretations in the spaces generated by the variables or by the coefficients. In variable space, any linear equation $\mathbf{a}^k\mathbf{x} = b_k$ defines a hyperplane $H_k = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^k\mathbf{x} = b_k\}$ orthogonal to $\mathbf{a}^k$ and MAX FLS$^=$ amounts to finding a point where the largest number of hyperplanes intersect. For nonstrict inequalities, each relation defines a closed halfspace $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^k\mathbf{x} \geq b_k\}$ and one seeks a point belonging to the largest number of halfspaces. In the particular case of homogeneous equations, one may also consider the coefficient space where each equation is associated to a point. According to this representation, MAX FLS$^=$ amounts to finding a hyperplane through the origin containing as many points as possible. For homogeneous systems with all inequalities in the same direction, one looks for a hyperplane in coefficient space whose induced halfspace contains the largest number of points associated to the coefficient vectors. Similar geometrical interpretations are also valid for MIN ULR variants as well as for strict inequalities, nonequalities and systems with mixed relations.

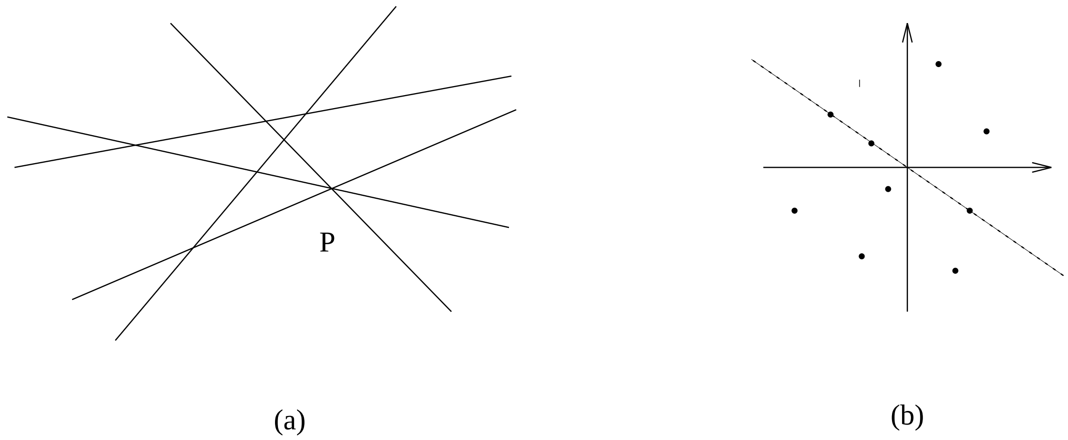


Figure 1.1: The two geometrical interpretations of MAX FLS and MIN ULR problems in variable and coefficient spaces. (a) Variable space. Each linear equation defines a hyperplane and one seeks a point where the largest number of hyperplanes intersect. (b) Coefficient space. Each homogeneous equation corresponds to its coefficient vector and one seeks a hyperplane through the origin that contains as many vectors as possible.

1.2.2 Political science

A special case of MAX FLS and MIN ULR with homogeneous inequalities arises in political science [94]. Suppose that a candidate, who knows the preferences of p voters on n issues, wants to determine a set of positions on the various issues which is likely to be supported by a majority of the voters. Geometrically, this problem can be formulated as follows. Consider the n-dimensional space where each dimension is associated to a relevant issue and the unit radius hypersphere S^{n-1} centered at the origin. Each voter is represented by the intersection of S^{n-1} with the vector whose components correspond to his positions with respect to the n issues. Given the set of p points $\{\mathbf{a}^k\}_{1\leq k\leq p}$ on the hypersphere associated to the p voters, the goal is to find a hemisphere of S^{n-1} containing as many points as possible. Clearly, vectors $\mathbf{x} \in \mathbb{R}^n$ that satisfy a maximum number of inequalities $\mathbf{a}^k\mathbf{x} \geq 0$, $1 \leq k \leq p$, correspond to the axes of hemispheres containing a maximum number of points $\mathbf{a}^k$. This problem, known as CLOSED HEMISPHERE, is a special case of MAX FLS with homogeneous inequalities and no pairs of collinear row vectors in A. The following variants have been considered in [94].

> CLOSED HEMISPHERE: Given a finite set $\{\mathbf{a}^k\}_{1\leq k\leq p}$ of n-dimensional vectors with rational components, find a vector $\mathbf{x} \in \mathbb{Q}^n$ with $\mathbf{x} \neq \mathbf{0}$ such that $\mathbf{a}^k\mathbf{x} \geq 0$ for as many $\mathbf{a}^k$ as possible.

> OPEN HEMISPHERE: Given a finite set $\{\mathbf{a}^k\}_{1\leq k\leq p}$ of n-dimensional vectors with rational components, find a vector $\mathbf{x} \in \mathbb{Q}^n$ such that $\mathbf{a}^k\mathbf{x} > 0$ for as many $\mathbf{a}^k$ as possible.

Note that these formulations are more general than the hemisphere interpretation since many points $\mathbf{a}^k$ can lie on the same ray from the origin.

Johnson and Preparata proved that OPEN HEMISPHERE as well as CLOSED HEMISPHERE are intractable, namely NP-hard, and they proposed a complete enumeration algorithm for tackling these special cases [94]. The method is based on a simple recursive strategy which can be easily sketched for CLOSED HEMISPHERE. According to the geometrical interpretation in variable space, the nonstrict inequalities in a CLOSED HEMISPHERE instance define a set of p hyperplanes $H_k = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^k\mathbf{x} = 0\}$, $1 \leq k \leq p$, passing through the origin. Such hyperplanes partition the variable space $\mathbb{R}^n$ into unbounded cones. Clearly, all vectors $\mathbf{x}$ in the same cone define hemispheres containing the same subset of points $\mathbf{a}^k$. This is in particular true for the points lying on the faces of the cone. Suppose that $\mathcal{C}$ is an optimal cone corresponding to hemispheres containing a maximum number of points. In order to find an optimal solution $\mathbf{x} \in \mathbb{R}^n$, we can restrict our attention to the faces of $\mathcal{C}$. Projecting the original points $\{\mathbf{a}^k\}_{1 \leq k \leq p}$ into one of the hyperplanes containing a nonempty face of $\mathcal{C}$ and solving the given $n-1$-dimensional Closed Hemisphere instance, provides a solution to the original problem. Since the appropriate hyperplanes are not known in advance, all $\mathbf{a}^k\mathbf{x} = 0$ with $1 \leq k \leq p$ must be considered. Thus an n-dimensional instance of Closed Hemisphere can be reduced to a set of at most n instances of the same type of problem in $n-1$ dimensions. The strategy can then be applied recursively to solve the auxiliary $(n-1)$-dimensional instances. Eventually, this leads to an exponential number of 1-dimensional Closed Hemisphere instances that are solvable in linear time, namely $O(p^n)$ instances. However, if an $O(p \log p)$ algorithm is used for $n = 2$ variables [94], the overall recursive method has an $O(p^{n-1} \log p)$ time-complexity for $n \geq 3$.

In a more general framework, we may want to attach a different importance to the solution of some relations than to that of others. This can be achieved by assigning a weight to each relation. The goal is then to find a solution which maximizes the total weight of the satisfied relations or minimizes the total weight of the unsatisfied ones. Johnson and Preparata's complete enumeration procedure is also applicable to the weighted and mixed variants containing strict as well as nonstrict inequalities.

1.2.3 Linear editing

Constrained versions naturally and frequently arise when modeling practical problems. Variants in which either a subset of relations is mandatory or the variables are restricted to take a finite set of values, deserve a particular attention. Often, mixed variants including different types of relations are also of interest.

As an example of mixed and constrained version of Max FLS and Min ULR, we mention a problem occurring in the field of linear numeric editing [77]. Suppose that a database is characterized by all vectors in a given polytope $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\}$, where A is a $(p \times n)$-matrix and $\mathbf{b}$ an n-dimensional vector. In other words, every vector in the database fulfills all inequalities of $A\mathbf{x} \leq \mathbf{b}$. For any vector $\tilde{\mathbf{x}}$ failing this set of p linear edits, we look for a smallest set of components of $\tilde{\mathbf{x}}$ that must be modified in order to obtain a vector in the database. Formally, the problem is to minimize the number of unsatisfied equations of the system

$$z_k = 0 \quad \text{for } 1 \leq k \leq p$$

involving the p auxiliary variables z_k, while satisfying the mandatory set of inequalities

$$A(\tilde{\mathbf{x}} + \mathbf{z}) \leq \mathbf{b}.$$

This is equivalent to minimize the number of satisfied nonequalities $z_k \neq 0$, $1 \leq k \leq p$, subject to the above inequality constraints.

By extending an algorithm proposed by Warmack and Gonzalez for solving a special case of Max FLS and Min ULR with homogeneous inequalities [163], Greer developed a general method for optimizing functions of systems of linear relations subject to linear constraints [77]. A function $f : \mathbb{R}^n \to \mathbb{R}$ is a *function of a system of linear relations* $\{\mathbf{a}^k\mathbf{x}\, \mathcal{R}_k\, b_k\}_{1\leq k\leq p}$ if and only if there exists a function $g : \{0,1\}^p \to \mathbb{R}$ such that for every $\mathbf{x} \in \mathbb{R}^n$

$$f(\mathbf{x}) = g(\mathbf{1}\{\mathbf{a}^1\mathbf{x}\, \mathcal{R}_1\, b_1\}, \ldots, \mathbf{1}\{\mathbf{a}^p\mathbf{x}\, \mathcal{R}_p\, b_p\}),$$

where $\mathcal{R}_k \in \{=, \geq, >, \neq\}$ and $\mathbf{1}\{y = z\}$ denotes the indicator function which is 1 if $y = z$ and 0 otherwise. The linear constraints can feature any of the four basic types of relational operators $=$, $\geq$, $>$ and $\neq$.

Although Greer's algorithm is designed to solve a particular type of problems without constraints and with only inequalities, a very large class of constrained or unconstrained problems is reducible to such a simple canonical form [77]. The definition of this class involves the notions of nondecreasing and nonincreasing variables. The kth variable of $f : \{0,1\}^p \to \mathbb{R}$ is said *nondecreasing* if for every choices of $\xi_1, \ldots, \xi_{k-1}, \xi_{k+1}, \ldots, \xi_p \in \{0,1\}$ we have

$$f(\xi_1, \ldots, \xi_{k-1}, 0, \xi_{k+1}, \ldots, \xi_p) \leq f(\xi_1, \ldots, \xi_{k-1}, 1, \xi_{k+1}, \ldots, \xi_p).$$

Similarly, the kth variable of f is nonincreasing if and only if the kth variable of $-f$ is nondecreasing and a function f is nondecreasing if and only if all its variables have that property. In [77] it was proved that any solution $\mathbf{x}$ which optimizes an arbitrary function $f(\mathbf{x})$ of a system of linear relations subject to linear constraints, can be obtained from a vector $\mathbf{y}$ which maximizes a corresponding positive function $f_c(\mathbf{1}\{\mathbf{a}^1\mathbf{y}\, \mathcal{R}_1\, 0\}, \ldots, \mathbf{1}\{\mathbf{a}^{p'}\mathbf{y}\, \mathcal{R}_{p'}\, 0\})$ with no nonincreasing variables and only homogeneous inequalities (i.e. $\mathcal{R}_k \in \{\geq, >\}$) and vice versa.

In fact, Greer's method can solve any problem in homogeneous canonical form with a nondecreasing objective function. Thus the linear editing problem as well as all Max FLS and Min ULR variants are good candidates. In weighted Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$, for instance, the objective is to minimize the function $\sum_{k=1}^{p} w_k\, \mathbf{1}\{\mathbf{a}^k\mathbf{x}\, \mathcal{R}\, b_k\}$ where the reals w_k, $1 \leq k \leq p$, denote the corresponding weights. As a consequence of the above-mentioned equivalence, any instance of weighted MAX FLS with mixed inequalities and real weights can be reduced to a particular instance with only positive weights [77].

Unlike Johnson and Preparata's method, the tree algorithm does not enumerate the values of the objective function on all the cones induced by the given set of hyperplanes but just on a subclass of them (called hills) that is known to contain all optimal solutions. The hills, which are typically far fewer than the cones, are enumerated using an appropriate tree. This method is more efficient than complete enumeration but still exponential in the worst-case. Namely, it has an $O(n \cdot p^n / 2^{n-1})$ time-complexity for instances with n variables and p relations [77].

1.2.4 Linear programming

While various problems in operations research and economics can be directly stated as MAX FLS or MIN ULR variants, a large class of examples indirectly arises in the important area of linear programming.

Linear programming (LP) is the field of mathematics concerned with maximizing or minimizing linear functions subject to a finite set of linear constraints. Any such problem with equalities or nonstrict inequalities constraints can be expressed in the standard form:

$$\begin{array}{rl} \min & \mathbf{c}\mathbf{x} \\ \text{s.c.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

where A denotes a $p \times n$ matrix, and $\mathbf{c}$, $\mathbf{x}$ as well as $\mathbf{b}$ are vectors of corresponding dimensions. Since the birth of LP in the 50's, a very wide range of apparently unrelated problems in management and science have been formulated as linear programs and solved using the famous simplex algorithm or interior point methods [73].

In recent years, a growing attention has been paid to linear programs with an infeasible set of constraints [76, 75]. Infeasibility frequently occurs when formulating or modifying large and complex models. Yet repairing programs with thousands of constraints is a difficult task that cannot be achieved by simple inspection. Due to the continual software and hardware advances, the real bottleneck in LP applications is more the correct formulation of large models than their effective solution. Dealing with infeasible programs turns out to be an important issue.

Various approaches have recently been investigated to diagnose infeasibility and try to resolve it. A first step along this line consists in isolating a subset of constraints that is infeasible. Several works have been devoted to finding *irreducibly inconsistent subsystems*, IIS for short [118, 68, 43]. Given a system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with $\mathcal{R} \in \{=, \geq, >\}$, an IIS is a subsystem that is infeasible but which becomes feasible if any constraint is dropped. Identifying an IIS helps the diagnosis by focusing on a small part of the linear system which partially accounts for infeasibility. Indeed, a feasible subsystem cannot contain all inequalities of any IIS. In [118] Van Loon suggested a pivoting method to enumerate all IIS's. Gleeson and Ryan made a substantial improvement by showing that finding all IIS's of an infeasible program can be reduced to enumerating all vertices of a related polytope [68]. Furthermore, Chinneck and Dravnieks developed several algorithms to identify IIS's [43].

Since the overall goal is to resolve infeasibility, the natural objective is to look for a maximum feasible subsystem of $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with $\mathcal{R} \in \{=, \geq, >\}$. Let $\{I_j\}_{1\leq j\leq s}$ denote the set of all IIS's of an infeasible system. In order to achieve feasibility, one must drop at least one constraint from each I_j. Finding a minimum set of inequalities to be deleted is equivalent to identifying a minimum cardinality cover for $\{I_j\}_{1\leq j\leq s}$ [68], that is to solving

$$
\begin{array}{lcc}
\min & \sum_{i=1}^{p} y_i & \\
\text{s.c.} & \sum_{i\in I_j} y_i \geq 1 & j = 1, \ldots, s \\
 & y_i \in \{0, 1\} & i = 1, \ldots, p
\end{array}
$$

where the variable y_i is 1 if the ith inequality is dropped and 0 otherwise. Clearly, deleting the constraints corresponding to any minimum cover yields a maximum feasible subsystem of the original system. Since the number of IIS's is typically exponential in the number of constraints, Parker and Ryan proposed an iterative heuristic which avoids complete enumeration of all IIS's. This is achieved by looking for partial covers and generating new IIS's only when they are needed [140, 75]. However, even if only a subset of IIS's are considered, this type of methods require a very high computational effort with respect to the size of the original system.

Resolving infeasibility of a linear program may lead either to a constrained variant of Max FLS with equations or to an unconstrained one with inequalities. If the nonnegativeness constraints are mandatory, one looks for a maximum feasible subsystem of $A\mathbf{x} = \mathbf{b}$ subject to $\mathbf{x} \geq \mathbf{0}$. Otherwise, each equation can be replaced by a couple of inequalities and the purpose is to achieve feasibility by deleting as few inequalities as possible, including the nonnegativeness ones.

Note that although Greer's algorithm can be used to resolve infeasibility of small-sized linear programs, it is not applicable to mixed-integer programs containing discrete variables.

1.2.5 Discriminant analysis

Another important variant of MAX FLS and MIN ULR arises in the field of discriminant analysis. The aim of classification is to assign observations to one of several predetermined classes based on a set of known attributes. Classification problems have a wide spectrum of applications in a number of disciplines including natural and social sciences, engineering, medicine and business [82].

Among the various classification rules that have been considered, linear discriminant ones are extensively used in practice because they are simple and easy to interpret [82]. In the two-class case, the goal is to determine an n-dimensional discriminant vector $\mathbf{w} \in \mathbb{R}^n$ and a scalar $w_0 \in \mathbb{R}$ such that an observation $\mathbf{a} \in \mathbb{R}^n$ is classified as belonging to group 1 if $\mathbf{aw} > w_0$ and to group 2 otherwise. Geometrically, if $\mathbf{w}$ is nontrivial, $(\mathbf{w}, w_0)$ defines a discriminant function that partitions the n-dimensional attribute space into two halfspaces induced by the hyperplane $H = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{wx} = w_0\}$. On one side we have the open halfspace $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{wx} > w_0\}$ and on the other side the closed one $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{wx} \leq w_0\}$. In general, given a set of p observations $\{\mathbf{a}^k\}_{1 \leq k \leq p}$ with the corresponding membership classes, the parameters $\mathbf{w}$ and w_0 are determined from the available data $\mathbf{a}^k$, $1 \leq k \leq p$, so that new observations with unknown class memberships are assigned to the class they are more likely to belong to. In the pattern recognition literature, linear discriminant functions are often referred to as *linear classifiers* [54].

The many methods for designing linear discriminant functions fall into two broad categories. On one hand, there are the statistical techniques which have interesting probabilistic properties but assume that the forms of the a priori and class-conditional distributions are either known or reliably estimable. Among them, the famous Bayesian procedures minimize the expected misclassification cost [128]. For instance, the celebrated Fisher's linear discriminant [60] is known to minimize the probability of misclassifications when the two classes have normal distributions with known means and equal variance-covariance matrices. Over the years a wide range of statistical tools have been proposed for discriminant analysis [128].

On the other hand, there are the nonparametric methods in which the forms of the discriminant functions are chosen a priori and the available data are used to estimate their parameters without making any assumption on the underlying distributions. Linear

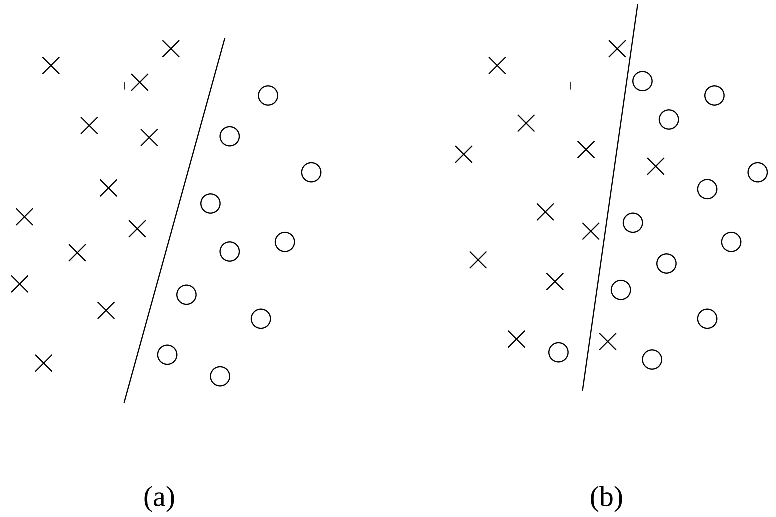


Figure 1.2: Examples of two-class linear classifiers with $n = 2$ attributes. Points in the first class are denoted by circles and those in the second one by crosses. The observations are linearly separable in (a) and nonlinearly separable in (b). In both cases an optimal hyperplane with a minimal number of misclassifications is represented.

discriminant functions are then designed by optimizing some distribution independent objective function that is a surrogate for minimizing the number of misclassifications, i.e. for finding a vector $\mathbf{w}$ and a scalar w_0 satisfying as many inequalities of the following system as possible

$$\begin{array}{ll} \mathbf{a}^k\mathbf{w} > w_0 & \text{if } \mathbf{a}^k \text{ belongs to the first class} \\ \mathbf{a}^k\mathbf{w} \leq w_0 & \text{if } \mathbf{a}^k \text{ belongs to the second class.} \end{array} \tag{1.2}$$

This problem, which is equivalent to maximizing the number of correct classifications, is obviously a special case of mixed Min ULR and Max FLS with strict and nonstrict inequalities.

During the last decade, many mathematical programming formulations have been studied for linear discriminant analysis (see [64, 110, 70, 35] as well as the included references). Two main types of approaches can be distinguished. In the first one, an effort is made to try to capture the goals of discriminant analysis into linear programs. The proposed models minimize, for instance, the maximum deviation or the sum of deviations from the misclassified observations to the current hyperplane. In order to avoid unacceptable or trivial solutions, increasingly sophisticated models have been devised [108, 35]. In the second approach, the problem of minimizing the number of misclassified observations is directly casted into mixed-integer programs involving real as well as binary variables [157, 110]. All these mathematical programming formulations require a slight modification of (1.2) because they cannot directly deal with strict inequalities [67, 108]. The experiments performed so far on simulated and real data indicate that these methods offer some advantages compared to statistical techniques, in particular under nonnormal data conditions (see the references in [157, 110]). The mixed-integer programming approach provides very good results and tends to outperform the LP formulations. However, it is computationally prohibitive even for small examples with a few attributes and a hundred observations. Since real classification problems have typically thousands of observations, all existing methods for solving mixed programs have a very limited range of application.

Recently, a genetic search strategy has been proposed for designing optimal linear classifiers [109]. The advantage of such a heuristic technique is that it can also take into account another important goal in classification, namely that of using as few attributes as possible. Since parsimony has a major impact on the performances for unknown data [16], one looks among the linear classifiers with a minimum number of misclassifications for those using a smallest subset of attributes. The problem of identifying a subset of most relevant attributes is well known in the statistical literature and is referred to as variable selection [128].

Not surprisingly, there exists a close relationship between the Bayesian approach and the minimization of the number of misclassifications. Greer proved that the true misclassification cost of a discriminant function that minimizes the empirical cost (i.e. actually measured on the available observations) converges almost surely to the minimum misclassification cost when the number of observations tends to infinity [77]. In particular, if the costs of misclassifying a first or second class observation are equal, Bayes empirical cost is equivalent to the number of misclassifications. If the two types of errors have different costs, minimizing the misclassification cost amounts to a weighted version of MIN ULR and MAX FLS.

The minimization of the number of misclassifications really permeates discriminant analysis. Even for statistical classifiers, performances are in general evaluated experimentally by computing the number of misclassifications on the given set of observations or on another test set.

All results pertaining to linear discrimination can of course be extended to the case of higher order polynomial classifiers, where the discriminant surfaces are any polynomial of degree smaller or equal to a fixed integer $d \geq 2$. This is simply due to the fact that polynomials are linear with respect to their coefficients. Note that the Bayesian approach naturally leads to quadratic discriminant functions when the two variance-covariance matrices are different [128].

1.2.6 Artificial neural networks

Machine learning and more precisely the rapidly growing area of artificial neural networks is the last important field of occurrence we shall mention. Since this is the topic of the second part of this dissertation, we will just sketch the context where a special case of Max FLS and Min ULR arises. This variant turns out to be closely related to discriminant analysis.

The term "artificial neural network" denotes a variety of systems composed of many simple and nonlinear computational elements that interact through some pattern of connections. The reader is referred to the textbooks [146, 111, 15, 85] for complementary introductions to the field. The various models differ in the topology, that is the number of units as well as their pattern of connections, and in the type of nonlinear operation performed by the computing units. The topology of a network can be represented as an oriented graph $G = (X, U)$ where the node set X corresponds to the units and the arc set U represents the connections between them. While the nodes with predecessors $V \subseteq X$ contribute to the overall computation by taking signals from their entering arcs and by calculating an output signal, those without predecessors are simple interface units used to supply inputs to the network.

The focus here is on the *linear threshold units* which were proposed in McCulloch and Pitts' pioneering work [127] and which are often referred to as *perceptrons* [130, 143]. As shown in Figure 1.3, a perceptron has n inputs x_j, $1 \leq j \leq n$, and a real *weight* w_j attached to each arc from the corresponding input x_j. Given an input vector $\mathbf{x}$, the output y is determined by computing the weighted sum of its n inputs x_j and by comparing the total input to a *threshold* w_0

$$y = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} w_j x_j > w_0 \\ -1 & \text{otherwise.} \end{cases} \tag{1.3}$$

Geometrically, the weight vector $\mathbf{w}$ and the threshold w_0 define a hyperplane $H = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{w}\mathbf{x} = w_0\}$ which divides the input space $\mathbb{R}^n$ into two disjoint halfspaces. A perceptron is thus equivalent to a linear classifier. Inputs may take real or discrete values, in the binary case we adopt the bipolar representation $\mathbb{B} = \{-1, 1\}$. The threshold is usually omitted because it can be considered as the weight associated to an additional input which is always set to -1. In continuous models, the step function is replaced by a continuous nonlinear function like the sigmoid or the hyperbolic tangent [85].

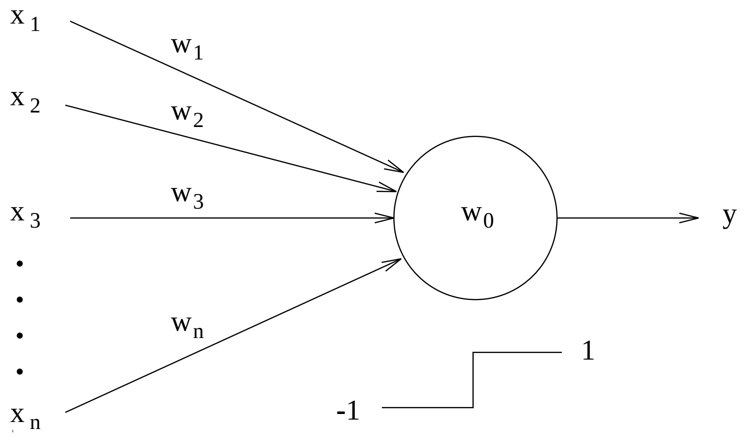


Figure 1.3: The building block of artificial neural networks. A single unit (perceptron) performs a linear threshold function by computing a weighted sum of its n inputs x_j with the corresponding weights w_j and by comparing the total input to a threshold w_0.

Among all possible models, *feedback* networks whose underlying graphs G contain at least one circuit are contrasted to *feedforward* networks whose graphs are circuit-free [85]. In this work we restrict our attention to the last ones.

Feedforward networks can be unfolded into a number of layers. The set $I = X - V$ of input nodes consists of all nodes without predecessors (first layer) and the set $O \subseteq V$ of output nodes is the set of nodes without successors (last layer). Information is processed in a natural way, layer by layer. When an input $\mathbf{a} \in \mathbb{R}^{|I|}$ is presented to the first layer nodes, the nodes within the same layer compute their outputs in parallel, from the second layer to the last one. The state of the nodes in O, denoted by $\mathbf{b} \in \mathbb{B}^{|O|}$ where $\mathbb{B} = \{-1, 1\}$, is then taken to be the corresponding output of the network.

Feedforward networks of linear threshold units are primarily used for associative tasks. Roughly speaking, training such a network amounts to storing a set of p input-output pairs of patterns $(\mathbf{a}^k, \mathbf{b}^k) \in \mathbb{R}^{|I|} \times \mathbb{B}^{|O|}$ with $1 \leq k \leq p$. In other words, the parameters of the network (weights and thresholds) must be tuned so that it provides an output close to $\mathbf{b}^k$ for each input vector $\mathbf{a}^k$. Usually, it is assumed that every input vector in the *task* $T = \{(\mathbf{a}^k, \mathbf{b}^k)\}_{1 \leq k \leq p} \subseteq \mathbb{R}^{|I|} \times \mathbb{B}^{|O|}$ is assigned to at most one output vector, so it can be considered as a function which is partially defined on the input space $\mathbb{R}^{|I|}$ and takes its values in the output space $\mathbb{B}^{|O|}$.

The simplest kind of feedforward networks are those with a single computing layer and a single unit. Training such a perceptron consists of, given a task $T = \{(\mathbf{a}^k, b^k)\}_{1\leq k\leq p} \subseteq \mathbb{R}^n \times \mathbb{B}$, finding a weight vector $\mathbf{w}$ that classifies all the $\mathbf{a}^k$ as well as possible. If the task is *linearly separable*, i.e. it can be performed correctly by a perceptron, this amounts to solving the corresponding linear system

$$\begin{aligned} \mathbf{a}^k\,\mathbf{w} > 0 \ \ \forall k \text{ such that } \ b^k = 1 \\ \mathbf{a}^k\,\mathbf{w} \leq 0 \ \ \forall k \text{ such that } \ b^k = -1. \end{aligned} \tag{1.4}$$

Whenever such a solution exists, the famous perceptron algorithm [143, 130] is guaranteed to yield a weight vector $\mathbf{w}$ that correctly classifies all $\mathbf{a}^k$ after a finite number of steps. Such a solution can also be found in polynomial time using an appropriate method for linear programming [73]. Unfortunately, the probability that a large-sized task with even binary inputs is linearly separable is extremely low because less than 2^{n^2} of the 2^{2^n} Boolean functions with $n \geq 4$ variables are linearly separable [2]. In the nonlinearly separable case, the best we can ask for is an *optimal* weight vector which maximizes the consistency, i.e. which correctly classifies as many input vectors as possible. As far as optimal solutions are concerned, this is equivalent to seeking a weight vector which minimizes the number of errors on the task. Thus a special case of MAX FLS and MIN ULR with mixed strict and nonstrict inequalities arises when training perceptrons, which are the building blocks of artificial neural networks.

Two extensions of the perceptron algorithm have been proposed to deal with nonlinearly separable tasks. The *pocket* procedure [65] converges for any task whose input vectors have integer components and the probability to get an optimal weight vector tends to 1 when the number of steps grows. The *thermal perceptron* algorithm [63] has no asymptotic convergence guarantee but provides better experimental results. Both of them will be discussed in detail in chapter 5. Implicit enumeration techniques like Greer's one could be used to find optimal weight vectors in finite time, but these exact procedures are only applicable to small-sized problems.

As will be seen, training single perceptrons is also of great interest for designing multilayer networks of linear threshold units. In a number of algorithms, the overall training problem is subdivided into that of training a sequence of single units. For instance, this is the case in the method of learning by choice of internal representations [79, 78, 147] or in the important class of constructive algorithms [129, 62, 155, 9]. Since the last strategies allocate units during the training phase, the size and performances of the constructed networks strongly depend on how well each unit performs the subtask it has been assigned.

Thus effective procedures are needed for maximizing single unit consistency.

1.3 Outline of the thesis

As seen in section 1.2, there is a considerable and growing interest in the class of MAX FLS and MIN ULR combinatorial problems. Previous and ongoing works include a number of algorithms for tackling the variants arising in a wide range of fields. A particular attention is devoted to versions with mixed types of relational operators, to weighted variants where a different importance can be assigned to each relation and to constrained ones where a subset of relations are mandatory or where the variables are restricted to take discrete values.

Surprisingly enough, before we started this work not much was known on the complexity of solving this widespread class of problems. For systems with real variables, the only result was Johnson and Preparata's proof that it is intractable to find an optimal solution to two special cases with homogeneous inequalities, namely OPEN and CLOSED HEMISPHERE [94]. However, nothing was known for the other types of relational operators (in particular equations) and for the important variants with additional weights or constraints. What is more, the approximability of these classes of optimization problems had not been studied even though approximate solutions are of great practical value.

A common feature of all known algorithms is that they are computationally prohibitive even for instances of moderate size. The fact that the best available techniques require a high computational burden suggests that these problems are not trivially solvable. However, it does not provide any definitive evidence on their inherent complexity, it just reflects the current state of the art. More efficient algorithms based on some yet unknown subtleties may exist.

The limited success encountered until recently in studying the approximability of intractable optimization problems partially explains the lack of results on the approximability of MAX FLS and MIN ULR variants. But, as we shall see, the progresses that have been made in complexity theory over the last few years open new research paths.

In the first part of this dissertation we consider the main variants of MAX FLS and MIN ULR arising in various fields within a unified framework and investigate their inherent complexity, independently from any particular algorithm.

Chapter 2 provides a brief overview of the computational complexity material that will be used. The main focus is on some of the new developments of approximability theory which are scattered throughout the literature.

In chapter 3 the complexity of the two complementary classes of problems, Max FLS and Min ULR, is studied in detail for any of the four types of relational operators $=$, $\geq$, $>$ and $\neq$. Besides the basic variants, the weighted and constrained versions are also considered. Since the interest is in approximate solutions as well as in optimal ones, we determine the complexity of solving these problems to optimality and, whenever they are intractable, we try to asses their degree of approximability as precisely as possible. This chapter is based on [7, 10, 12].

The second part of the thesis is devoted to the special variants of Max FLS and Min ULR arising in the rapidly growing field of artificial neural networks. As already mentioned in section 1.2.6, attention is restricted to feedforward networks of linear threshold units.

In chapter 4 we discuss the implications of our complexity results on the difficulty of training this important class of networks. Among others, we disprove the "folk" theorem stating that training single-layer networks is easy while it becomes intractable as soon as there are hidden units [6]. Our discussion goes beyond artificial neural networks and leads to an extension of the famous PAC model of learning from examples that makes it more realistic. The fundamental issue of designing close-to-minimum size networks is also addressed with a special focus on nonzero weights instead of computing units. In particular, we derive a strong bound on the approximability of the design problem even for single perceptrons, which are the building blocks of neural networks.

In chapter 5 we present and analyze several variants of the famous perceptron algorithm that are well behaved for nonlinearly separable tasks [8]. The focus is on consistency maximization. First we describe some simple extensions of the thermal rule that lead to considerable improvements. Then we propose a new probabilistic variant of the perceptron algorithm. Unlike the thermal perceptron, this procedure is guaranteed to find an optimal weight vector with probability approaching 1 when the number of iterations tends to infinity. Comparative results for various types of tasks are reported.

In chapter 6 feedforward neural networks of threshold units with hidden layers are considered. Three constructive methods are proposed which allocate units during the training phase and train them using single perceptron procedures [9]. These methods are compared to other existing strategies for designing multilayer networks through a set of experiments with simulated and real data. Our interest is in simple techniques that require local information and are reasonable candidates for hardware implementation.

In the conclusion we summarize the main aspects of this work and suggest several paths for future research.

The definitions of all computational problems mentioned throughout the thesis are listed in the Appendix.

Chapter 2

Preliminaries

In this chapter we briefly introduce the definitions and material about computational complexity which will be used in the first part of this dissertation.

2.1 Computational complexity

The purpose of *computational complexity* is to study the computational cost of solving certain problems independently from any particular algorithm. The inherent difficulty of a problem is the least cost of all possible algorithms that would solve it. This cost is measured in terms of the amount of resources needed to compute a solution, namely *time* and *space*. Of course the amount of space is bounded by the time used and the overall resources required depend upon the size of the instance.

The origin of computational complexity goes back to the seminal work by Hartmanis and Stearns [83] where the concept of time complexity was first formalized. After some computational models and cost measures were investigated, a major advance was the birth of the NP-completeness theory [46, 102]. Showing that a problem is NP-complete makes extremely unlikely the existence of an efficient algorithm providing an optimal solution for every possible instance. In this framework, algorithms requiring a polynomial amount of time in the size of the instance are contrasted to those requiring exponential time. While the last ones are clearly impractical, the former ones are considered to be efficient. Assuming that the fundamental conjecture P$\neq$ NP is true, NP-complete problems cannot have polynomial time algorithms.

Although the NP-completeness theory was primarily developed to investigate the apparent intractability of various optimization problems, the class NP was defined in terms of decision problems with yes or no answers [66]. Thus, for any optimization problem a decision version is derived by imposing a bound on the value of the objective function. Since optimization problems are at least as hard as their decision versions, showing that the last ones are intractable directly implies the hardness of finding optimal solutions to the former ones.

In principle, every combinatorial optimization problem can be solved in finite time by enumerating all feasible solutions. But the number of feasible solutions being typically exponential, enumerative procedures are practically hopeless even for instances of very moderate size.

The NP-completeness theory, based on the concepts of Turing machines and polynomial time reductions, provided a robust framework for studying decision problems and for classifying optimization problems according to the complexity of finding optimal solutions. However, NP-hardness results are just a first step towards a real understanding of the complexity of difficult optimization problems. The central question is whether there exist efficient algorithms which always provide good approximate solutions. This is a fundamental issue from a theoretical as well as a practical point of view. Indeed optimal solutions are rarely required and near-optimal ones, which are guaranteed to be a fixed percentage away from the optimum, often suffice.

In [93] Johnson started to classify optimization problems according to their approximability properties. Later, a number of efficient approximate algorithms were devised for some intractable problems [137, 87] and a few isolated nonapproximability results were established. While the traveling salesman problem without triangular inequality was proved to be as hard to approximate as to solve optimally [149], the maximum independent set in a graph was shown to be either extremely easy or very hard to approximate [66].

The various attempts to extend the NP-completeness analysis were limited by the fact that the standard tools are not suited to study the complexity of approximating optimization problems. In particular, the simple polynomial time reductions are not strong enough to preserve approximations. Moreover, the approximability properties of many problems which are equivalent to solve optimally can differ enormously. This is even true for intractable optimization problems whose NP-complete decision versions are reducible to each other. While some of them, like the knapsack problem [91], have very efficient approximation algorithms others, like the traveling salesman, are very hard to approximate.

This type of results rise a number of important questions. Why do similar problems have different approximability properties ? How to define meaningful classes of problems with similar approximability ? Do stronger kinds of reducibility which preserve approximability in some way exist?

Over the last few years there have been new substantial progresses in the study of the approximability of NP-hard optimization problems. Various complexity classes have been introduced in order to group problems according to their approximation properties [66, 49]. Some of them characterize the approximability of problems directly from their structural properties without involving any particular approximation measure [138, 23]. Different reductions which preserve approximability in various ways have also been introduced and used to compare the approximability of some optimization problems [22, 141, 135, 49]. These works provided a much-needed framework to investigate the difficulty of finding approximate solutions.

Further significant advances have recently been triggered by a set of important results obtained in the area of interactive proofs [24]. Exploiting a clever connection between interactive proof systems and the approximability of several functions, strong bounds have been derived on the hardness of finding good approximate solutions to some famous optimization problems like maximum independent set, minimum graph coloring and minimum dominating set [20, 120, 32]. These remarkable results, which answered some longstanding open questions, have important consequences on the approximability of other optimization problems and open new research paths [24].

For any given optimization problem, the ultimate goal is to determine the best performance ratio that is achievable in polynomial time. Therefore, positive as well as negative results are needed. While better approximation algorithms guarantee as good as possible solutions, hardness results provide lower bounds on the problem's approximability. Thus upper and lower bounds should be matched in order to reveal the exact degree of approximability.

It should be noted that all the above developments of complexity theory are based on a worst-case analysis. Since the focus is on efficient algorithms which are guaranteed to provide an optimal solution for any possible instance, the difficulty of a problem depends on the hardest instance. Although an average-case analysis would be important from a practical point of view, the assumptions that must be made about the input distributions are very difficult to justify. This partially accounts for the fact that the various attempts undertaken so far led to unresolved controversies [114].

In the sequel we assume basic knowledge of computational complexity as well as graph theory and focus on the recent developments in approximability theory. In spite of the interest of these new and remarkable advances, we shall just cover the concepts and results that will be used in the next chapters. The reader is referred to the classical textbooks [66, 137] for the background material and to [24] for a review of some recent developments based on the interactive proofs connection. While a large account on approximation algorithms is gathered in [132], a number of definitions and results on the approximability of NP-hard optimization problems are listed in [47, 99].

2.2 Optimization problems

Since we are concerned with optimization problems, the following formal definition is in order.

Definition 2.2.1 [49, 47] An NP optimization (NPO) problem is a four-tuple $\Pi = (\mathcal{I}_\Pi, S_\Pi, f_\Pi, goal_\Pi)$ where

- $\mathcal{I}_\Pi$ is the set of the *instances* of Π and it is recognizable in polynomial time.
- $S_\Pi(I)$ is the set of *feasible solutions* of the instance $I \in \mathcal{I}_\Pi$. These solutions are short, i.e. there exists a polynomial p such that for any solution $x \in S_\Pi(I)$ we have $|x| \leq p(|I|)$. Furthermore, it is decidable in polynomial time whether, for any I and for any x such that $|x| \leq p(|I|)$, $x \in S_\Pi(I)$.
- $f_\Pi : \mathcal{I}_\Pi \times \Sigma^* \rightarrow \mathbb{N}$, the *objective function*, is a polynomial time computable function. $f_\Pi(I, x)$ is defined only for $x \in S_\Pi(I)$.
- $goal_\Pi \in \{\max, \min\}$ tells if Π is a *maximization* or a *minimization* problem.

Solving an optimization problem Π given an instance $I \in \mathcal{I}_\Pi$ amounts to find a $x \in S_\Pi(I)$ such that $f_\Pi(I, x)$ is optimum, that is as large as possible if $goal_\Pi = \max$ and as small as possible if $goal_\Pi = \min$. The optimal value of f_Π over $S_\Pi(I)$ is denoted by $opt_\Pi(I)$.

In order to determine the difficulty of finding an optimal solution of an NPO problem we consider the associated decision version.

Definition 2.2.2 For any NPO problem $\Pi = (\mathcal{I}_\Pi, S_\Pi, f_\Pi, goal_\Pi)$, the corresponding *decision problem* is to decide, for any given $I \in \mathcal{I}_\Pi$ and integer K, whether there exists a feasible solution $x \in S_\Pi(I)$ such that $f_\Pi(I, x) \geq K$ when Π is a maximization problem (and $f_\Pi(I, x) \leq K$ when Π is a minimization problem).

By definition 2.2.1, it is easily verified that all decision problems associated to NPO problems are in the class NP.

Definition 2.2.3 A problem Π is NP-*hard* if any problem Π' in NP can be reduced to Π in polynomial time. In other words, Π is at least as hard as any Π' in NP. Π is NP-*complete* if $\Pi \in$ NP and Π is NP-hard.

Thus, unless P=NP, it is extremely unlikely that such a problem can be solved optimally in time polynomial in the input size.

The above definition of NPO problems is broad enough to encompass all known optimization problems whose decision versions are in NP. The class of all NPO problems is called NPO.

Example 2.2.1 Max FLS and Min ULR with any type of relational operators are NPO problems. Consider the case of Max FLS with strict inequalities. According to Khachian [106], the predicate which indicates whether such a linear system is feasible can be evaluated in polynomial time. Let this predicate be denoted by $\mathbf{LSF}(I)$ where $I = \{(a_{i1}, \ldots, a_{in}, b_i) \ : \ i \in \{1 \ldots p\}\} \subset \mathbb{Q}^{n+1}$ is a set of inequalities such as $a_{i1}x_1 + \cdots + a_{in}x_n > b_i$. Then we have

- $\mathcal{I}_\Pi$ the set of all finite systems of strict linear inequalities with rational coefficients,
- $S_\Pi(I) = \{I' \subseteq I \mid \mathbf{LSF}(I')\}$ the set of all feasible subsystems of I,
- $f_\Pi(I, Y) = |Y|$ the size of the feasible subsystem $Y \in S_\Pi(I)$ of $I \in \mathcal{I}_\Pi$.

The size of an instance $I \in \mathcal{I}_\Pi$ of Max FLS depends on the number of relations p, the number of variables n and the size of the rational coefficients.

In the sequel we shall often restrict our attention to a particular class of NPO problems whose objective function is bounded by a polynomial in the size of the instance.

Definition 2.2.4 [37, 113] An NPO problem Π is *polynomially bounded* if there is a polynomial $p : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$ such that

$$opt_\Pi(I) \leq p(|I|) \quad \forall I \in \mathcal{I}_\Pi.$$

The class of all polynomially bounded NPO problems is called NPO PB and the subclass of all polynomially bounded maximization (minimization) problems Max PB (Min PB). Thus NPO PB = Max PB $\cup$ Min PB.

MAX FLS and MIN ULR are both in NPO PB since their objective function is the number of satisfied or unsatisfied relations. A famous NPO PB problem we shall consider is maximum independent set.

> MAX IND SET: Given an undirected graph $G = (V, E)$, find a maximum independent set of nodes, i.e. a set of nodes that are not pairwise adjacent.

Of course many NPO problems do not belong to NPO PB. This is, for instance, the case of the traveling salesman problem where, given a set of cities and the distances between them, one seeks a shortest tour which visits each city exactly once.

Note that all decision or optimization problems mentioned in this thesis are listed and defined in the Appendix.

2.3 Approximation algorithms and schemes

Formally, approximating an optimization problem Π given an instance $I \in \mathcal{I}_\Pi$ amounts to finding any $x \in S_\Pi(I)$. The quality of the approximation depends of course on the relation between $f_\Pi(I, x)$ and $opt_\Pi(I)$. Two equivalent measures are extensively used for this purpose: the relative error and the performance ratio.

Due to the lack of symmetry of the relative error between maximization and minimization problems, the last measure is often preferred.

Definition 2.3.1 The *performance ratio* of a feasible solution with respect to the optimum of an NPO problem Π is defined as

$$R_\Pi(I, x) = \begin{cases} opt_\Pi(I)/f_\Pi(I, x) & \text{if } opt_\Pi = \max, \\ f_\Pi(I, x)/opt_\Pi(I) & \text{if } opt_\Pi = \min, \end{cases}$$

where $I \in \mathcal{I}_\Pi$ and $x \in S_\Pi(I)$.

By definition, R_Π is always greater or equal to 1.

The performance of an approximation algorithm is then defined as the worst-case of its performance ratio over all inputs of a given size.

Definition 2.3.2 An NPO problem Π *can be approximated within* c for a constant c if there exists a polynomial time algorithm $\mathcal{A}$ such that for all instances $I \in \mathcal{I}_\Pi$, $\mathcal{A}(I) \in S_\Pi(I)$ and $R_\Pi(I, \mathcal{A}(I)) \leq c$. More generally, Π *can be approximated within* $p(n)$ for a function $p : \mathbb{Z}^+ \to \mathbb{R}^+$ if there exists a polynomial time algorithm $\mathcal{A}$ such that for every $n \in \mathbb{Z}^+$ and for all instances $I \in \mathcal{I}_\Pi$ with $|I| = n$ we have that $\mathcal{A}(I) \in S_\Pi(I)$ and $R_\Pi(I, \mathcal{A}(I)) \leq p(n)$.

A problem which can be approximated within a constant c is said to be *c-approximable*. Obviously 1-approximable problems are polynomially solvable.

As previously mentioned, the approximability properties of NPO problems can differ considerably. Some of them can be approximated in polynomial time within any constant factor $c > 1$.

Definition 2.3.3 An NPO problem has a *polynomial time approximation scheme* (PTAS) if there exists a polynomial time algorithm $\mathcal{A}(I, \varepsilon)$ which provides, for all instances $I \in \mathcal{I}_\Pi$ and all $\varepsilon > 0$, a solution $x \in S_\Pi(I)$ such that $R_\Pi(I, x) \leq 1 + \varepsilon$. For any fixed ε, the time complexity of $\mathcal{A}$ is polynomial in the instance size.

MAX IND SET restricted to planar graphs has, for example, a PTAS with a time complexity $O(n \cdot 8^{1/\varepsilon}/\varepsilon)$, where n is the number of nodes in the graph [25].

In some cases, the time complexity of an approximate algorithm is not only polynomial in the size of the input but also in $1/\varepsilon$.

Definition 2.3.4 An NPO problem has a *fully polynomial time approximation scheme* (FPTAS) if there exists a PTAS whose running time is polynomial in the input size as well as in $1/\varepsilon$.

Among all NPO problems with an NP-complete decision version, those which have a FPTAS are the easiest ones and can be very well approximated. The famous maximum knapsack problem has a FPTAS [91].

Two simple conditions involving the concept of strong NP-completeness make the existence of a FPTAS extremely unlikely. A decision problem is said to be *strongly* NP-*complete* if it is intractable even when the time complexity is measured with respect to the size of the largest numbers occurring in the instance (i.e. with respect to unary coding of the data).

Proposition 2.3.1 [66] *Let* Π *be an* NPO *problem satisfying the following properties:*

i) the decision version of Π *is strongly* NP*-complete,*

ii) opt_Π *is bounded from above by a polynomial in the size of the instance as well as in the size of its largest number.*

Then Π *cannot have a* FPTAS *unless* P = NP.

For polynomially bounded NPO problems, strong NP-completeness of the decision version is enough to rule out a FPTAS because the second condition is always verified.

Of course many interesting problems, like the traveling salesman, are much harder to approximate than the knapsack problem, namely they cannot be approximated within any constant factor.

Thus, even if the NPO problems with NP-complete decision versions are all equally hard to solve optimally they may have very different approximability properties. The main reason for this difference is that the simple polynomial time reductions between the corresponding decision problems may not preserve the objective function or the degree of approximability.

Example 2.3.1 Consider MAX IND SET and its complementary problem minimum node cover.

> MIN NODE COVER: Given an undirected graph $G = (V, E)$, find a set of nodes $V' \subseteq V$ of minimum cardinality which is incident to every edge, i.e. for each $[v_1, v_2] \in E$ at least one extremity is in V'.

Since a subset of nodes is independent if and only if its complement is a node cover, there exists a node cover of size s if and only if there exists an independent set of size $n - s$, where n is the number of nodes in the graph. MAX IND SET and MIN NODE COVER having complementary decision versions, they are equally hard to solve optimally. But, while MIN NODE COVER can be approximated within 2 [26], we shall see in section 2.6 that MAX IND SET is not approximable within any constant factor.

The same remark holds for the complementary problems MAX FLS and MIN ULR with the same type of linear relations. Although they are equivalent to solve optimally, their approximability may differ considerably.

2.4 Approximation preserving reductions

In order to relate optimization problems according to their approximation properties, we need reductions that are stronger than the simple polynomial time ones. Moreover, they must be designed for optimization problems to avoid the usual and artificial procedure of converting them to decision problems.

Various type of reductions which preserve different approximation measures have been proposed over the last years [37, 135, 49, 136]. For instance, the P-reduction and F-reduction were defined to preserve PTAS and respectively FPTAS. In other words, if an NPO problem Π P-reduces (F-reduces) to another NPO problem Π which has a PTAS (FPTAS) then Π has also a PTAS (FPTAS).

Among the reductions which preserve approximation within constants, the linear reduction (L-reduction) introduced by Papadimitriou and Yannakakis is the most restrictive one [99]. Moreover, it can easily be used to show that a given problem is at least as hard to approximate as another one.

Definition 2.4.1 [138] Given two NPO problems Π and Π', Π *L-reduces* to Π' if there exist two polynomial time computable functions t_1, t_2 and two constants α, $\beta > 0$ such that

i) t_1 reduces every instance I of $\mathcal{I}_\Pi$ to an instance I' of $\mathcal{I}_{\Pi'}$ with the property that $opt_{\Pi'}(I') \leq \alpha \cdot opt_\Pi(I)$,

ii) t_2 maps solutions x' of I' with objective value $f_{\Pi'}(I', x') = c'$ to solutions x of I with $f_\Pi(I, x) = c$ such that $|opt_\Pi(I) - c| \leq \beta\ |opt_{\Pi'}(I') - c'|$.

If Π L-reduces to Π' we denote it by $\Pi \leq_L^p \Pi'$, where p indicates that it is a polynomial time reduction.

The composition of L-reductions is clearly an L-reduction. The way an L-reduction preserves approximation within constants depends on the values of α and β.

Proposition 2.4.1 [138] *If* $\Pi \leq_L^p \Pi'$ *with constants* α *and* β *and there is a polynomial time approximation algorithm for* Π' *with worst-case relative error* ε*, then there is a polynomial time approximation algorithm for* Π *with worst-case relative error* $\alpha\beta\varepsilon$*.*

An L-reduction with $\alpha = \beta = 1$ is a *cost preserving transformation* in the sense that it preserves the values of the objective function.

Definition 2.4.1 can be slightly modified in order to render it impervious to scaling of the objective function, that is to make scaled-up versions of the same problems interreducible. Therefore the left hand side of the inequality in ii) should be divided by $opt_\Pi(I)$ and the right hand side by $opt_{\Pi'}(I')$ [107].

Unfortunately, relative error preserving reductions, like the L-reduction, are not appropriate to relate the approximability of NPO problems which cannot be approximated within any constant. In such cases, the approximability is described by a function $u(n)$ where n is the instance size. Since a reduction between two such problems Π and Π' can associate small instances of Π to much larger instances of Π', a good approximation algorithm for Π' may lead to a much worse approximation algorithm for Π. Thus the *size amplification* has a major impact on how the approximability changes through a reduction.

The following type of reduction, which takes into account the size amplification, is particularly suited for relating problems that cannot be approximated within any constant.

Definition 2.4.2 [100] Given two NPO problems Π and Π', a four-tuple $t = (t_1, t_2, a(n), c)$ is an *S-reduction with size amplification* $a(n)$ from Π to Π' if

i) t_1, t_2 are polynomial time computable functions, $a(n)$ is a monotonously increasing positive function and c is a positive constant,

ii) $t_1 : \mathcal{I}_\Pi \to \mathcal{I}_{\Pi'}$ is such that $\forall I \in \mathcal{I}_\Pi$ and $\forall x \in S_{\Pi'}(t_1(I))$ we have $t_2(I, x) \in S_\Pi(I)$,

iii) $\forall I \in \mathcal{I}_\Pi$ and $\forall x \in S_{\Pi'}(t_1(I))$, we have $R_\Pi(I, t_2(I, x)) \leq c \cdot R_{\Pi'}(t_1(I), x)$,

iv) $\forall I \in \mathcal{I}_\Pi$, $|t_1(I)| \leq a(|I|)$.

If Π S-reduces to Π' we denote it by $\Pi \leq_S^p \Pi'$.

If the size amplification is $O(n)$, where n is the size of the instance, we say that the reduction is *without size amplification.*

The composition of S-reductions is clearly an S-reduction.

Proposition 2.4.2 [100] *If* $\Pi \leq_S^p \Pi'$ *with size amplification* $a(n)$ *and* Π' *can be approximated within some monotonously increasing function* $u(n)$ *in the size of the input instance, then* Π *can be approximated within* $c \cdot u(a(n))$.

It is easily verified that for constant and polylogarithmic[1] approximable problems the S-reduction preserves approximation within a constant for any polynomial size amplification. For n^ε-approximable problems with $\varepsilon > 0$, the S-reduction preserves approximation within a constant just for $O(n)$ size amplification.

An S-reduction with $c = 1$ is a *cost preserving transformation.*

[1]A polylogarithmic function in n is a polynomial function in $\log n$.

2.5 Some approximation classes

In order to compare and classify NPO problems according to their approximability properties, a number of approximation classes have been introduced. The most common ones are listed and discussed in [99].

For each class we are interested in the hardest problems, that is those to which every problem in the given class can be reduced.

Definition 2.5.1 Given an NPO problem Π and a class C, Π is C*-hard* if every $\Pi' \in C$ can be L-reduced to F. Π is C*-complete* if Π belongs to C and is C-hard.

As far as approximation is concerned, C-complete problems are good representatives of the class C.

2.5.1 Classes defined using approximability measures

The most natural way to define approximation classes is according to the polynomial time approximability degree of the optimization problems contained within the class. The existence of polynomial time approximation schemes gives rise to the classes PTAS and FPTAS containing all NPO problems which have, respectively, a PTAS or a FPTAS.

Another important distinction is in order between the problems which are approximable within at least some constant c and those which cannot be approximated within any $c > 1$.

Definition 2.5.2 APX is the class of all NPO problems which can be approximated within some constant factor.

As already mentioned, MIN NODE COVER is approximable within 2. A number of other interesting problems turn out to be in APX. The two following examples shall be considered in chapter 3.

> MAX SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

> MAX CUT: Given an undirected graph $G = (V, E)$, find a set of nodes $V' \subseteq V$ such that the number of edges $[v_i, v_j] \in E$ with $v_i \in V'$ and $v_j \in V - V'$ is as large as possible.

Although for more than two decades the best-known approximation algorithms were embarrassingly simple, some substantial progresses have been made recently. In particular, it was shown that MAX SAT is approximable within 4/3 [167] and MAX CUT within 1.13823 [72].

As already pointed out, many problems, like the traveling salesman, do not belong to APX.

Remark 2.5.1 Clearly we have

$$\text{PO} \subseteq \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{NPO}$$

where PO is the class of all NPO problems which can be solved optimally in polynomial time. These inclusions are strict provided P $\neq$ NP.

An important class that lies in the middle of this range, between PTAS and APX, plays a fundamental role.

2.5.2 MAX SNP: a class defined by logical formulae

The class MAX SNP (for Strict NP), which was introduced by Papadimitriou and Yannakakis [138], is based on Fagin's characterization of NP using logical formulae [57]. Unlike for FPTAS, PTAS or APX, the focus is on the structure of the problem definition without considering any approximation measure.

Definition 2.5.3 An NPO problem Π is in the class MAX SNP if its objective function can be expressed as

$$opt_{\Pi}(I) = \max_{S} |\{x \ : \ \Phi(I, S, x)\}|$$

where Φ is a quantifier-free first-order formula, I an instance of Π and S a solution described as a finite structure.

In other words, MAX SNP is the class of all problems whose objective function can be expressed as the size of a set which is specified using the simplest type of Boolean formula.

Example 2.5.1 MAX IND SET restricted to graphs with bounded degree, namely MAX IND SET-B, is in MAX SNP. The graph $G = (V, E)$ is represented as a (B+1)-ary adjacency relation A. For every node v there is a tuple $(v, v_1, \ldots, v_B)$ where $v_1, \ldots, v_B$ is a list of its neighbors. If a node has less than B neighbors, we just make enough copies of one of them. Then the objective function of MAX IND SET-B can be expressed as

$$\max_S |\{(v, v_1, \ldots, v_B) \ : \ v \in S \wedge v_1 \notin S \wedge \ldots \wedge v_B \notin S\}|$$

where S should be viewed as a set of nodes.

On the contrary, the unrestricted MAX IND SET does not belong to MAX SNP.

Although MAX SNP is defined as a class of maximization problems, it contains also some minimization problems like MIN NODE COVER restricted to graphs of bounded degree [99].

MAX SNP is particularly interesting because it allows to characterize the approximability of several problems simply from the structure of their definition.

Proposition 2.5.1 [138] *Every problem in* MAX SNP *can be approximated within some constant.*

Several well-known problems turn out to be complete for MAX SNP under L-reductions. The archetype of MAX SNP-complete problem is:

> MAX 3SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most three literals, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

Other examples include MAX CUT and MAX IND SET-B. According to proposition 2.5.1, if any such problem has a PTAS then all those in MAX SNP would have a PTAS.

The question whether MAX SNP-hard problems are approximable within arbitrarily small factors remained open until recently. In [20] Arora, Lund, Motwani, Sudan and Szegedy provided a negative answer by exploiting a clever connection with the seemingly unrelated area of interactive proofs.

Theorem 2.5.2 [20] *For every* MAX SNP*-hard problem there exists a constant $c > 1$ such that finding an approximate solution within c is* NP*-hard.*

This rules out the existence of a PTAS for all MAX SNP-hard problems, including the many that are not in MAX SNP like, for instance, MAX IND SET and LONGEST PATH in a graph [139]. In fact, these two problems have an interesting self-amplification property: if they can be approximated within some constant then they can be approximated within every constant. In other words, either these problems have a PTAS or they do not belong to APX. Hence they are not approximable within any constant factor.

According to theorem 2.5.2, showing that a problem is MAX SNP-complete describes its approximability quite well: it can be approximated within a constant but not within every constant. Therefore we have: PTAS $\subseteq$ MAX SNP $\subseteq$ APX.

Estimating the threshold c turns out to be difficult because it depends on the properties of the interactive proof system used in the reduction. Although the initial value was very small, it has by now been improved to $c = 74/73$ for MAX 3SAT by delving into the underlying proof systems [33].

Note that the class hinted in [138] is the closure of MAX SNP under L-reductions, denoted by $\overline{\text{MAX SNP}}$ [99]. Since the *closure* of a class C is the set of all problems which can be reduced to a problem in C, we have MAX SNP $\subseteq$ $\overline{\text{MAX SNP}}$.

A number of other syntactically-defined classes have been introduced and studied in the literature [138, 112, 113]. Unlike computationally-defined classes such as FPTAS, PTAS and APX, these syntactic classes are well-suited to find natural complete problems. However, the approximation properties of their problems are not as well understood. As we shall see in the next section, an important step has recently been done to reconcile these two complementary views of approximation classes.

2.6 Towards a hierarchy of approximation classes

The above-defined approximation classes give rise to the following sequence of inclusions:

$$\text{PO} \subseteq \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{MAX SNP} \subseteq \text{APX} \subseteq \text{NPO},$$

where all inclusions are strict provided $\text{P} \neq \text{NP}$. As previously mentioned, these classes contain maximization as well as minimization problems.

The range of approximability of NP-hard optimization problems stretches from problems which can be approximated within every constant in polynomial time, i.e. that have a PTAS, to problems which cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$, unless $\text{P} = \text{NP}$, where n is the size of the input instance.

Even polynomially bounded maximization problems can be very hard to approximate, namely not in APX. A famous example is MAX IND SET. Although this problem has been known to be NP-hard for 20 years, until recently there was no explanation as to why all attempts to find good approximate solutions had failed. But in [20] Arora, Lund,

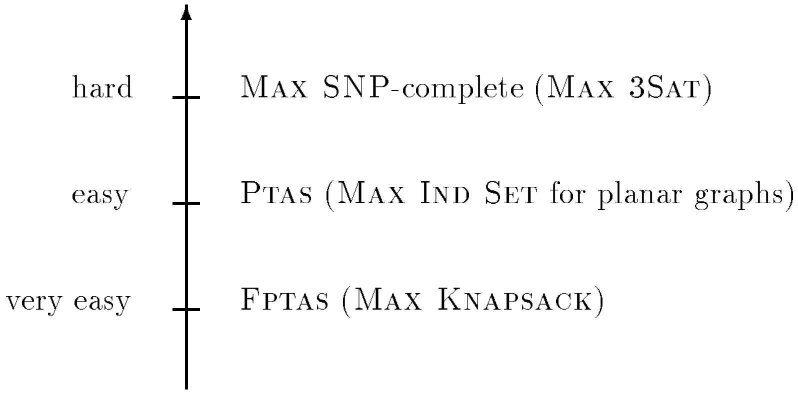


Figure 2.1: Different approximability degrees of NP-hard optimization problems which can be approximated within some constant (which belong to Apx).

Motwani, Sudan and Szegedy provided an answer using the powerful interactive proofs connection.

Theorem 2.6.1 [20] *Assuming* P $\neq$ NP*,* Max Ind Set *cannot be approximated within a factor of* n^ε *for some* $\varepsilon > 0$*, where* n *is the number of nodes in the input graph.*

The best estimation of ε is by now equal to 1/6 [33]. By strengthening the assumption P $\neq$ NP, namely by assuming that NP $\neq$ co-RP[2], the above nonapproximability result holds for $\varepsilon = 1/5$.

If there is an approximation preserving reduction from Max Ind Set to an NPO problem Π we say that Π is Max Ind Set-*hard*, in other words it is at least as hard to approximate as the maximum independent set problem.

Definition 2.6.1 A polynomially bounded NPO problem Π is said to be NPO-*complete* if every NPO PB problem can be reduced to Π using an approximation preserving reduction.

Several natural maximization problems like Max Ones are known to be complete for Max PB [36].

> Max Ones: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of clauses with at most 3 literals, find a truth assignment for X that satisfies every clause and that contains as many true variables as possible.

[2]The definition of co-RP is recalled in section 2.7.

In fact, since Crescenzi, Kann and Trevisan have recently proved that Max PB = Min PB [48], these problems turn out to be NPO PB-complete.

Due to the asymmetry between minimization and maximization problems, the former problems do not necessarily have the same degrees of approximability as the last ones. Several important problems like Min Dominating Set and Min Graph Colouring lie in the middle of the approximability range, on the harder side of Apx.

> Min Dominating Set: Given an undirected graph $G = (V, E)$, find a dominating set $V' \subseteq V$ of minimum cardinality, i.e. for all $v \in V - V'$ there exists $v' \in V'$ such that $[v, v'] \in E$.

> Min Graph Colouring: Given an undirected graph $G = (V, E)$, color the nodes of G with as few colors as possible so that no two adjacent nodes receive the same color.

Recently Lund and Yannakakis established strong nonapproximability bounds for these problems using the interactive proof connection.

Theorem 2.6.2 [120] *Assuming* P $\neq$ NP*,* Min Graph Colouring *cannot be approximated within n^ε for some $\varepsilon > 0$, where n is the number of nodes in the graph.*

In [32] Bellare, Goldwasser, Lund and Russell improved the result for Min Dominating Set and several closely related problems such as minimum set cover and hitting set.

Theorem 2.6.3 [32] Min Dominating Set *cannot be approximated within any constant factor unless* P = NP*, and within $c \log(n)$ for any $c < 1/8$ unless* NP $\subseteq$ DTIME($n^{\log\log n}$)*, where n is the number of nodes in the graph.*

DTIME($T(n)$) denotes the class of all decision problems which can be solved in an amount of time bounded by $T(n)$, where n is the size of the input instance. Although it is believed that NP $\not\subseteq$ DTIME($n^{\log\log n}$), this is not as likely as P $\neq$ NP. The same results hold for the set of closely related problems mentioned in [120].

If there is an approximation preserving reduction from Min Dominating Set to an NPO problem Π we say that Π is Min Dominating Set-*hard*, in other words it is at least as hard to approximate as the minimum dominating set problem.

As for maximization problems, several polynomially bounded minimization problems, including the well-known Min Ind Dom Set, are complete for Min PB [100] and hence for NPO PB-complete.

> MIN IND DOM SET: Given an undirected graph $G = (V, E)$, find an independent set V' of minimum cardinality which dominates all nodes of G.

By improving a construction proposed in [92], Halldórsson established the following strong nonapproximability result.

Theorem 2.6.4 [81] *Assuming* P $\neq$ NP, MIN IND DOM SET *cannot be approximated within a factor of* $n^{1-\varepsilon}$ *for any* $\varepsilon > 0$*, where* n *is the number of nodes in the graph.*

As a matter of fact, this bound is still valid if n is the sum of the number of nodes and edges in the graph. By definition, all NPO PB-complete problems are as hard to approximate as MIN IND DOM SET. It is worth pointing out that such a nonapproximability bound makes the existence of any nontrivial and general approximation algorithm extremely unlikely.

Similar nonapproximability bounds are also valid for NPO PB-complete maximization problems [48]. For instance, the following variant of MAX ONES

> MAX DONES: Given two finite sets X, Z of variables and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most 3 literals, find a truth assignment for X and Z that satisfies every clause in $\mathcal{C}$ and contains as many Z variables set to TRUE as possible.

cannot be approximated within $|Z|^{1-\varepsilon}$ for any $\varepsilon > 0$ unless P = NP, where $|Z|$ is the number of distinguished variables.

The above-mentioned set of results give strong lower bounds on the best performance ratio that can be guaranteed by any polynomial time algorithm. These negative results provide substantial advances in the study of the dividing line between what is efficiently approximable and what is not. But to complete the picture, one should try to close the gap between the approximation factors which are NP-hard and those which are known to be achievable in polynomial time. So far, only very few optimization problems have a well-characterized approximability.

An interesting example is related to MAX FLS with equations. Given a set of degree-2 equations over a finite field $GF(q)$, where q is a small prime, the objective is to satisfy as many equations as possible. Håstad, Phillips and Safra have shown in [84] that if the polynomials do not contain any squares as monomials this problem can be approximated within $q^2/(q-1)$ in polynomial time (for fixed q) but not within $q - \varepsilon$ for any $\varepsilon > 0$ unless P = NP. Thus, the gap between the factor to which this problem is approximable and that to which it is NP-hard, is equal to $q/(q-1)$, which tends to 1 for large values of q.

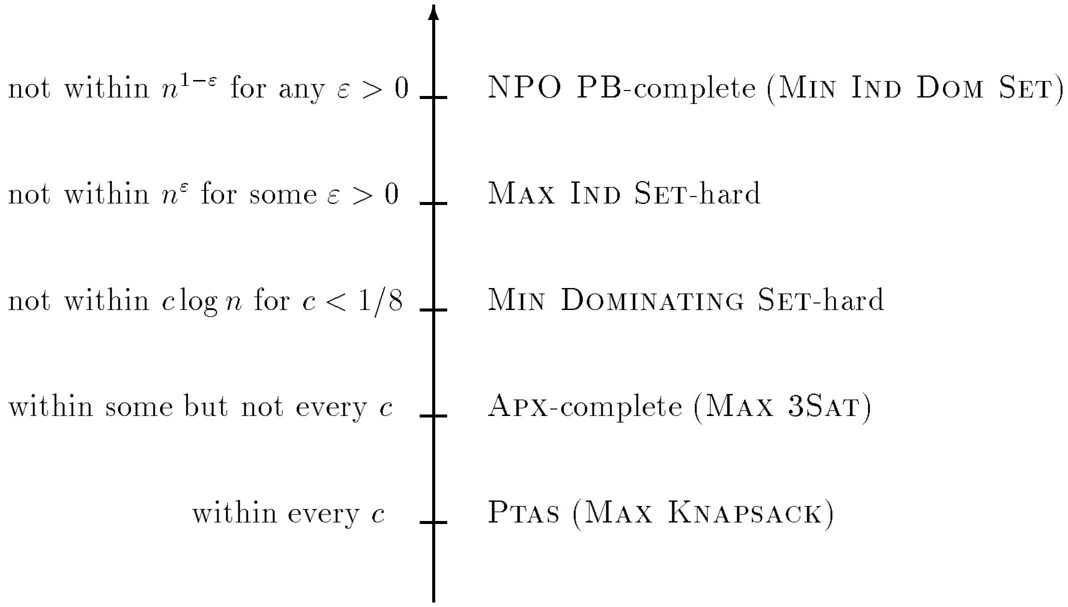


Figure 2.2: Different approximability degrees of NP-hard optimization problems.

The hardness results mentioned in this section lay the basis for a hierarchy of approximation classes. However, in a true approximation hierarchy problems from lower classes should be reducible to complete problems in higher classes using an appropriate type of reduction.

Lately, an interesting step has been done in this direction [107]. The purpose of Khanna, Motwani, Sudan and Vazirani was to try to reconcile the syntactic and computational views of approximation classes. By using the tightened definition of L-reduction mentioned in section 2.5 to define the closure class $\overline{\text{Max SNP}}$, they have shown that $\overline{\text{Max SNP}}$= Apx. This is the first result which provides simultaneously a syntactic and computational view of an approximation class, namely the class of all problems which are approximable within some constant factor. Thus Max SNP-hard problems that belong to Apx, like Max 3Sat, turn out to be Apx-*complete* (see also [21]). Moreover, according to theorem 2.5.2, Apx-hard problems cannot be approximated within every constant in polynomial time.

In the same paper the authors claim similar results for other approximation classes. For instance, they state that Max Ind Set is complete for the class *poly*-Apx of all problems which are approximable within polynomial ratios. This claim sounds quite surprising since it would imply that Max Ind Set is in fact NPO PB-complete. If it turns out to be true, this would prove the existence of a hierarchy of approximation classes with natural complete problems for each class (e.g. Max 3Sat for Apx and Max Ind Set for *poly-*Apx). Nevertheless, this result cannot be considered valid as long as the proof is not published.

The purpose of chapter 3 is to investigate the approximability status of the different Max FLS and Min ULR variants and to locate them within the known approximation classes. We shall see that the approximability of apparently similar and closely related problems can differ enormously.

2.7 Randomized algorithms: basic definitions

We conclude this chapter with a few basic definitions related to randomized computation. The class co-RP has already been mentioned in section 2.6.

The following definitions are stated in terms of decision problems instead of formal languages. The reader is referred to [103] for a unified introduction to randomized algorithms.

A *randomized algorithm* is one which receives, in addition to the input string describing a particular instance of the problem at hand, also a string of random bits ρ.

Definition 2.7.1 A randomized algorithm $\mathcal{A}$ is said to solve a decision problem Π if and only if for every instance $I \in \mathcal{I}_\Pi$

i) $\mathcal{A}$ confirms *yes* instances with probability strictly greater than 1/2,

ii) $\mathcal{A}$ produces a *no* answer for *no* instances regardless of the string of random bits.

Such an algorithm is called a *Monte Carlo algorithm.*

Although $\mathcal{A}$ gives no definitive result in case of *no* answer, a long sequence of trials with *no* answers provides a good evidence for a *no* instance.

We are of course interested in Monte Carlo algorithms with a time-complexity which is bounded by a polynomial in the input size. ρ can be viewed as an infinite string of which only a finite number of components are used. Clearly, the 1/2 upper bound on the probability of error can be reduced to $1/2^t$ by applying t-times the Monte Carlo algorithm recursively.

Definition 2.7.2 RP is the class of all decision problems which can be solved by polynomial time Monte Carlo algorithms and co-RP is the class of all decision problems whose complement is in RP.

Since the part of ρ that is used can be considered as a polynomially bounded certificate [66], we have $\mathrm{P} \subseteq \mathrm{RP} \subseteq \mathrm{NP}$.

Within computational complexity theory, the fundamental conjecture states that NP-complete problems cannot be solved in polynomial time, i.e. $\mathrm{P} \neq \mathrm{NP}$. Although this question seems currently out of reach, these two complexity classes are widely believed to be different. Within the same framework, it is also conjectured that all decision problems in NP cannot be solved by polynomial time Monte Carlo algorithms, i.e. $\mathrm{RP} \neq \mathrm{NP}$.

Chapter 3

The complexity of Max FLS and Min ULR

In this chapter we investigate the computational complexity of the two classes of problems Max FLS and Min ULR with any of the four types of relational operators $=$, $\geq$, $>$ and $\neq$. Although Max FLS and Min ULR with the same type of relations are equivalent to solve optimally, they can have, as pointed out in chapter 2, very different approximability properties.

We will establish the complexity of solving these problems to optimality and, whenever they are intractable, we shall study their approximability. A number of important variants of the basic Max FLS and Min ULR are also considered. In particular, the weighted versions whose relations may have different importances and the constrained variants where a subset of relations must be satisfied or where the variables are restricted to take bounded discrete values.

We shall prove that Max FLS with $=$, $\geq$ or $>$ relations is NP-hard even when restricted to a subclass of homogeneous linear systems, whereas it can be solved in polynomial time for $\neq$ relations with real coefficients. The various intractable versions of Max FLS and Min ULR belong to different approximation classes depending on the type of relations and the additional constraints. We will show that the range of approximability of Max FLS variants is quite different from that of Min ULR variants, even for pairs of complementary problems with the same type of linear relations.

The chapter is subdivided in four parts. The first section is devoted to the complexity of solving Max FLS and Min ULR to optimality. The second section is concerned with the approximability of Max FLS variants while the third one deals with that of Min

ULR variants. Finally, the last section contains a few general comments on the difference between the approximability of the two classes of complementary problems.

3.1 The hardness of solving Max FLS and Min ULR optimally

In this section we focus on the basic problems Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ that are equivalent to the corresponding versions of Min ULR. Note that Max FLS$^{\neq}$ is trivially solvable because any system with $\neq$ relational operators is feasible. Indeed, for any finite set of hyperplanes associated with such a set of linear relations there exists a vector $\mathbf{x} \in \mathbb{R}^n$ that does not belong to any of them.

In order to determine the complexity of solving Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ to optimality, we consider the corresponding decision versions that are no harder than the original optimization problems. Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A, a p-dimensional vector $\mathbf{b}$ both with rational coefficients, and a positive integer K such that $1 \leq K \leq p$, does there exist a solution $\mathbf{x} \in \mathbb{Q}^n$ satisfying at least K relations of the system?

In the homogeneous versions of Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq\}$ we are not interested in the trivial solutions where all variables occurring in the satisfied relations are zero.

As previously mentioned, homogeneous Max FLS$^{=}$ can be viewed geometrically as follows. Given a set of p points in $\mathbb{R}^n$, find a hyperplane passing through the origin and containing the largest number of points.

Theorem 3.1.1 Max FLS$^{=}$ *is* NP*-hard even when restricted to homogeneous systems with ternary coefficients in* $\{-1, 0, 1\}$.

Proof We proceed by polynomial time reduction from the following NP-complete problem.

> Exact 3-sets Cover: Given a set S with $|S| = 3q$ elements and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets $C_j \subseteq S$ with $|C_j| = 3$ for $1 \leq j \leq m$, does $\mathcal{C}$ contain an exact cover, i.e. $\mathcal{C}' \subseteq \mathcal{C}$ such that each element s_i of S belongs to exactly one element of $\mathcal{C}'$?

Let $(S, \mathcal{C})$ be an arbitrary instance of Exact 3-sets Cover. We will construct a particular instance of Max FLS$^{=}$ denoted by $(A, \mathbf{b})$ such that the answer to the former one is affirmative if and only if the answer to the latter one is also affirmative.

The idea is to construct a system containing one variable x_j for each subset $C_j \in \mathcal{C}$, $1 \leq j \leq m$, and at least one equation for each element s_i of S, $1 \leq i \leq 3q$. We consider the following set of equations

$$\sum_{j=1}^{|\mathcal{C}|} a_{ij} x_j = 1 \text{ for } i = 1, \ldots, 3q, \tag{3.1}$$

where $a_{ij} = 1$ if the element $s_i \in C_j$ and $a_{ij} = 0$ otherwise, as well as the additional ones

$$x_j = 1 \quad \text{for} \quad j = 1, \ldots, m, \tag{3.2}$$

$$x_j = 0 \quad \text{for} \quad j = 1, \ldots, m. \tag{3.3}$$

Moreover, we set $K = 3q + m$. Clearly, K is equal to the largest number of equations that can be simultaneously satisfied.

Given any exact cover $\mathcal{C}' \subseteq \mathcal{C}$ of $(S, \mathcal{C})$, the vector $\mathbf{x}$ defined by

$$x_j = \begin{cases} 1 & \text{if } C_j \in \mathcal{C}' \\ 0 & \text{otherwise} \end{cases}$$

satisfies all equations of type (3.1) and exactly m equations of types (3.2)–(3.3). Hence $\mathbf{x}$ fulfills $K = 3q + m$ equations.

Conversely, suppose that we have a solution $\mathbf{x}$ which satisfies at least $K = 3q + m$ equations of $(A, \mathbf{b})$. By construction, this implies that $\mathbf{x}$ fulfills all equations of type (3.1) and m equations of types (3.2)–(3.3). Thus the subset $\mathcal{C}' \subseteq \mathcal{C}$ defined by $C_j \in \mathcal{C}'$ if and only if $x_j = 1$ is an exact cover of $(S, \mathcal{C})$.

The reduction can easily be extended to homogeneous MAX FLS$^=$. We just need to add a new variable x_{m+1} with $a_{i,m+1} = -b_i$ for all i, $1 \leq i \leq p$, and to observe that in any non trivial solution $\mathbf{x}$ we must have $x_{m+1} \neq 0$. Indeed, $x_{m+1} = 0$ would necessarily imply $x_j = 0$ for all j, $1 \leq j \leq m$. □

The question arises as to whether the problem is still intractable for systems with bipolar coefficients in $\{-1, 1\}$.

Corollary 3.1.2 MAX FLS$^=$ *remains* NP-*hard for homogeneous systems with bipolar coefficients.*

PROOF We extend the reduction of theorem 3.1.1 by using a duplication technique which allows to reduce systems with ternary coefficients in $\{-1, 0, 1\}$ to systems with bipolar coefficients in $\{-1,1\}$. The idea is to replace each variable x_j, $1 \leq j \leq n$, by 2 variables that are forced to be equal and that have only bipolar coefficients.

Consider an arbitrary instance of homogeneous MAX FLS$^=$ derived from an instance of EXACT 3-SETS COVER. Without loss of generality, we can assume that $a_{ij} \in \{-2,0,2\}$. This simple multiplication by a factor 2 does not affect the set of solutions but makes all coefficients even. Since the absolute value of a_{ij} is either 0 or 2, we can construct a system with bipolar coefficients that is equivalent to (3.1)–(3.3) by duplicating the variables and adding new equations.

Suppose that n different variables occur in (3.1)–(3.3). We associate with each equation $\mathbf{ax} = 0$ with n variables an equation $\tilde{\mathbf{a}}\mathbf{y} = 0$ with $2n$ variables. The coefficient vector $\tilde{\mathbf{a}}$ is determined as follows:

$$\tilde{a}_j = \begin{cases} 1 & \text{if } a_{\lceil \frac{j}{2} \rceil} = 2 \text{ or } a_{\lceil \frac{j}{2} \rceil} = 0 \text{ and } j \text{ is odd} \\ -1 & \text{if } a_{\lceil \frac{j}{2} \rceil} = -2 \text{ or } a_{\lceil \frac{j}{2} \rceil} = 0 \text{ and } j \text{ is even,} \end{cases}$$

where $1 \leq j \leq 2n$. This defines a mapping from $\{-2,0,2\}^n$ into $\{-1,1\}^{2n}$ which associates to each component a_i of $\mathbf{a}$ the two components $\tilde{a}_{2i-1}$ and $\tilde{a}_{2i}$ of $\tilde{\mathbf{a}}$ such that $a_i = \tilde{a}_{2i-1} + \tilde{a}_{2i}$ for all i, $1 \leq i \leq n$. For any $\mathbf{x} \in \mathbb{R}^n$ satisfying $\mathbf{ax} = 0$, the vector $\mathbf{y} \in \mathbb{R}^{2n}$ given by $y_j = x_{\lceil \frac{j}{2} \rceil}$ satisfies the corresponding equation $\tilde{\mathbf{a}}\mathbf{y} = 0$. Furthermore, if $y_{2i} = y_{2i-1}$ for $1 \leq i \leq n$ and $\tilde{\mathbf{a}}\mathbf{y} = 0$ then the vector $\mathbf{x}$ given by $x_i = y_{2i-1}$, $1 \leq i \leq n$, is a solution of $\mathbf{ax} = 0$. Thus, in order to construct an equivalent system with only $\{-1,1\}$ coefficients we must add new equations which eliminate the n additional degrees of freedom that have been introduced by mapping the original n-dimensional problem into the $2n$-dimensional one. In particular, we should ensure that $y_{2i} = y_{2i-1}$ for all $1 \leq i \leq n$. This can be achieved by satisfying simultaneously the 2 following homogeneous equations with bipolar coefficients:

$$\begin{aligned} y_{2i} - y_{2i-1} + \textstyle\sum_{1 \leq l \leq n,\ l \neq i} (y_{2l} - y_{2l-1}) &= 0 \\ y_{2i} - y_{2i-1} - \textstyle\sum_{1 \leq l \leq n,\ l \neq i} (y_{2l} - y_{2l-1}) &= 0. \end{aligned} \tag{3.4}$$

Indeed, $v + w = 0$ and $v - w = 0$ necessarily imply $v = w = 0$. Many equivalent equations of type (3.4) are needed in order to guarantee that $y_{2i} = y_{2i-1}$ for each i, $1 \leq i \leq n$. For each constraint $y_{2i} = y_{2i-1}$, we include a number of pairs of equations (3.4) that is larger than the number of equations of type (3.1)–(3.3). This can always be done by selecting the coefficients $\tilde{a}_{2l}$, $\tilde{a}_{2l-1} \in \{-1,1\}$ occurring in

$$\sum_{1 \leq l \leq n,\ l \neq i} \tilde{a}_{2l} y_{2l} + \tilde{a}_{2l-1} y_{2l-1}$$

in different ways (any choice with $\tilde{a}_{2l} = -\tilde{a}_{2l-1}$ is adequate).

It is worth noting that this general technique can be adapted to reduce any system of equations whose coefficients are restricted to take a finite number of values to an equivalent system with only bipolar coefficients. □

This result has an immediate consequence on the complexity of MAX FLS$^{\geq}$ which is stronger than that established by Johnson and Preparata for CLOSED HEMISPHERE [94]. Indeed, their reduction holds for systems with integer coefficients in $\{-4,-2,-1,0,1,4\}$.

Corollary 3.1.3 MAX FLS$^{\geq}$ *is* NP-*hard for homogeneous systems with bipolar coefficients.*

PROOF By simple polynomial time reduction from MAX FLS$^{=}$. Let $(A,\mathbf{0})$ be an arbitrary instance of homogeneous MAX FLS$^{=}$. For each equation $\mathbf{a}^i\mathbf{x}=0$ where $\mathbf{a}^i$ denotes the ith row of A, $1\leq i\leq p$, we consider the two inequalities $\mathbf{a}^i\mathbf{x}\geq 0$ and $-\mathbf{a}^i\mathbf{x}\geq 0$. Clearly, there exists a vector satisfying at least K equations of $(A,\mathbf{b})$ if and only if there exists a solution satisfying at least $p+K$ inequalities of the corresponding system with $2p$ inequalities. □

This is also true for systems with strict inequalities.

Theorem 3.1.4 MAX FLS$^{>}$ *is* NP-*hard for homogeneous systems with bipolar coefficients.*

PROOF The proof is by polynomial time reduction from the NP-complete problem MAX IND SET.

Let $G=(V,E)$ be an arbitrary instance of MAX IND SET. For each edge $[v_i,v_j]\in E$ we construct the following inequality

$$x_i+x_j<0 \tag{3.5}$$

and for each node $v_i\in V$ we consider the inequality

$$x_i>0. \tag{3.6}$$

Thus we have a system with $|E|+|V|$ strict inequalities.

We claim that the given graph G contains an independent set I of size s if and only if there exists a solution $\mathbf{x}$ satisfying all inequalities of type (3.5) and s inequalities of type (3.6).

Given an independent set $I \subseteq V$ of size s, the solution obtained by setting

$$x_i = \begin{cases} 1 & \text{if } v_i \in I \\ -2 & \text{otherwise} \end{cases}$$

satisfies all edge-inequalities of type (3.5) and all the node-inequalities of type (3.6) corresponding to a node $v_i \in I$. Moreover the $|V| - s$ inequalities associated with $v_i \notin I$ are not fulfilled because $x_i < 0$.

Conversely, given a solution $\mathbf{x}$ which satisfies all edge-inequalities and s node-inequalities, we consider the set $I \subseteq V$ containing all nodes whose node-inequality is fulfilled. The size of I is obviously equal to s. We verify by contradiction that I is an independent set. If I contains two adjacent nodes v_i and v_j, then we must have, on one hand, $x_i > 0$ and $x_j > 0$ and, on the other hand, $x_i + x_j < 0$ which is impossible. Hence I is an independent set of cardinality s.

In order to complete the proof we must make sure that all edge-inequalities are satisfied. This can be achieved by adding $|V|$ equivalent copies of each one of them, and in particular by multiplying each edge-inequality by different integer factors $f \in \{2, \ldots, |V|+1\}$. Thus we have a system with $(|V|+1)|E|$ inequalities of the first type and $|V|$ of the second one. Clearly, the given graph G contains an independent set I of size s if and only if there exists a solution $\mathbf{x}$ satisfying $(|V|+1)|E| + s$ strict inequalities.

This polynomial time reduction can be extended to MAX FLS$^{>}$ with bipolar coefficients by applying Carver's transposition theorem [151]. According to this result, a homogeneous system $A\mathbf{x} < \mathbf{0}$ is feasible if and only if $\mathbf{y} = \mathbf{0}$ is the unique solution of

$$\begin{aligned} A^t\mathbf{y} &= \mathbf{0} \\ \mathbf{y} &\geq \mathbf{0}. \end{aligned} \tag{3.7}$$

Thus any instance of MAX FLS$^{>}$ can be associated with such a system. Using the technique described in the proof of corollary 3.1.2, it is then possible to construct, for each system (3.7) with integer coefficients taking their values in $\{-(|V|+1), \ldots, 0, \ldots, |V|+1\}$, an equivalent system with only bipolar coefficients. It suffices to add a large enough number of appropriate equations forcing the new variables associated with any original variable to be equal. $\square$

Consequently, MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ is intractable not only when the points corresponding to the rows of A lie on the n-dimensional hypersphere but also when they belong to the n-dimensional hypercube. In such instances no relation differs with another one by a multiplicative factor.

Since these problems are NP-hard for bipolar coefficients, they turn out to be strongly NP-hard, i.e. intractable even with respect to unary coding of the data. According to proposition 2.3.1, they do not have a fully polynomial time approximation scheme unless P = NP.

Before turning to the approximability of MAX FLS and MIN ULR variants, we mention two special cases that are polynomially solvable.

Proposition 3.1.5 MAX FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *restricted to systems with a constant number of variables* n *can be solved in polynomial time.*

PROOF Greer's algorithm works for any type of linear relations and has an $O(n \cdot p^n/2^{n-1})$ time-complexity, where p denotes the number of relations and n the number of variables [77]. $\square$

Note that if the number of relations p is constant the problem is trivial because all subsystems can be checked in time $O(n)$.

Definition 3.1.1 An instance $(A, \mathbf{b})$ of MAX FLS$^{\mathcal{R}}$ is said to be *matroidal* if there exists a matroid (S, T) [151] such that each element in the ground set S corresponds to a relation of $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$, $|S| = p$ and each independent subset in T corresponds to a feasible subsystem.

Proposition 3.1.6 MAX FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *restricted to matroidal instances can be solved in polynomial time.*

PROOF Since all maximal independent sets (with respect to inclusion) have the same cardinality, we pick one relation at random and consider the other ones sequentially. At each step of this greedy procedure, we can check whether the additional relation is consistent with the current system using a polynomial time algorithm for linear programming. If it is consistent, the new relation is included in the partial feasible subsystem, otherwise it is discarded. Such a procedure is guaranteed to produce a maximum feasible subsystem. $\square$

3.2 The hardness of approximating Max FLS

In this section we investigate the approximability of Max FLS$^{\mathcal{R}}$ with any of the four types of relations $\mathcal{R} \in \{=, \geq, >, \neq\}$. After studying the basic versions in section 3.2.1, we analyze a number of important variants. Section 3.2.2 is devoted to Max FLS$^{=}$ restricted to finite fields. Various other versions with equations and inequalities are considered in sections 3.2.3 and 3.2.4. First we focus on the weighted problems where the goal is to maximize the total weight of the satisfied relations and on the constrained versions where the purpose is to find a solution which satisfies all mandatory relations and as many optional ones as possible. Then we consider the particular cases in which the variables are restricted to take a finite number of discrete values. In section 3.2.5 the overall structure underlying the various results is discussed and open questions are mentioned.

3.2.1 Basic Max FLS

Since the previous NP-hardness results make extremely unlikely the existence of polynomial time algorithms for solving the three basic versions of Max FLS to optimality, we are interested in efficient algorithms that are guaranteed to provide near-optimal solutions.

We will show that Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ and integer coefficients cannot be approximated within every constant unless P = NP. The proofs are by L-reductions from the following Max SNP-complete and hence Apx-complete problem [138].

> Max 2Sat: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most 2 literals in each clause, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

In homogeneous Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq\}$ we are not interested in the trivial solutions where all variables occurring in the satisfied relations are zero. However, forbidding such solutions does not suffice because there might be other trivial solutions with almost all variables equal to zero except a few ones that occur in just a few relations. In order to rule out these uninteresting situations for the equality and nonstrict inequality cases, we only consider solutions where the variable (variables) occurring in the largest number of satisfied relations is (are) nonzero.

Theorem 3.2.1 Max FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq\}$ *is* Apx*-hard even when restricted to homogeneous systems with integer coefficients in* $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ *and no pairs of identical relations.*

PROOF We first consider MAX FLS$^=$. Let $(X, \mathcal{C})$ with $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an arbitrary instance of MAX 2SAT. For each clause C_i, $1 \le i \le m$, containing two variables x_{j_1} and x_{j_2} we construct the following equations:

$$
\begin{aligned}
a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} &= 2 && (3.8)\\
a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} &= 0 && (3.9)\\
x_{j_1},\ x_{j_2} &= 1 && (3.10)\\
x_{j_1},\ x_{j_2} &= -1 && (3.11)
\end{aligned}
$$

where $a_{ij} = 1$ if x_j occurs positively in C_i and $a_{ij} = -1$ if x_j occurs negatively. Thus we have a system with $6m$ equations.

Given a truth assignment that satisfies s clauses of the MAX 2SAT instance, we immediately get a solution $\mathbf{x}$ which satisfies $2m + s$ equations of the MAX FLS$^=$ instance. This is simply achieved by setting the variables x_j to 1 or -1 depending on whether the corresponding Boolean variable is TRUE or FALSE in the assignment.

Consider any solution $\mathbf{x}$ of the MAX FLS$^=$ instance. For each i, $1 \le i \le m$, at most 3 equations can be simultaneously satisfied: at most one of (3.8)–(3.9) and at most one of (3.10)–(3.11) for each of the two variables. If any component of $\mathbf{x}$ is neither 1 nor -1, we can set it to 1 without decreasing the number of satisfied equations. In other words, we can suppose that any solution $\mathbf{x}$ has bipolar components.

Consequently, we have a correspondence between solutions of the MAX 2SAT instance satisfying s clauses and solutions of the MAX FLS$^=$ instance fulfilling $2m + s$ equations. Since opt[MAX FLS$^=$] $\le 3m$ and since there exists an algorithm providing a truth assignment that satisfies at least $\lceil \frac{m}{2} \rceil$ of the clauses in any MAX 2SAT instance (see for example [93]), we have opt[MAX FLS$^=$] $\le 6 \cdot$ opt[MAX 2SAT]. Hence all conditions for an L-reduction are fulfilled.

This L-reduction can be extended to homogeneous systems with bounded coefficients in $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ and no pairs of identical equations. For each clause C_i, $1 \le i \le m$, we add a new variable $x_{|X|+i}$ and we consider the following equations:

$$
\begin{aligned}
a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} - 2x_{|X|+i} &= 0 && (3.12)\\
a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} &= 0 && (3.13)\\
x_{j_1} - x_{|X|+i} &= 0 && (3.14)\\
x_{j_1} + x_{|X|+i} &= 0 && (3.15)\\
x_{j_2} - x_{|X|+i} &= 0 && (3.16)
\end{aligned}
$$

$$\begin{aligned}
x_{j_2} + x_{|X|+i} &= 0 && (3.17)\\
f x_{|X|+i} - f x_{|X|+m+1} &= 0 \quad \text{for all } f \in \{1,2,3,4\} && (3.18)\\
-f x_{|X|+i} + f x_{|X|+m+1} &= 0 \quad \text{for all } f \in \{1,2,3,4\} && (3.19)
\end{aligned}$$

where the coefficients a_{ij} are defined as above. Thus we have a system with $14m$ equations and $|X|+m+1$ variables. Here the correspondence is between solutions of the MAX 2SAT instance satisfying s clauses and feasible solutions $\mathbf{x}$ fulfilling $10m+s$ equations of this homogeneous system.

Given a truth assignment which satisfies s clauses, the solution obtained by setting $x_j = \pm 1$ for $1 \le j \le |X|$ depending on whether the corresponding Boolean variable is TRUE or FALSE and $x_{|X|+i} = 1$ for $1 \le i \le m+1$ satisfies at least $10m+s$ equations. Conversely, consider an arbitrary solution $\mathbf{x}$ satisfying $10m+s$ equations. By definition of homogeneous MAX FLS$^=$, we know that $x_{|X|+m+1}$ is nonzero for $m \ge 4$ because it occurs in at least $4m+s$ satisfied equations while any x_j with $1 \le j \le |X|$ occurs in at most $4m$ equations and any $x_{|X|+i}$ with $1 \le i \le m$ in at most 13. If $x_{|X|+i} \ne x_{|X|+m+1}$ for any i, $1 \le i \le m$, we can set it to $x_{|X|+m+1}$ without decreasing the number of satisfied equations. According to the same argument, any x_j with $1 \le j \le |X|$ that is neither $x_{|X|+m+1}$ nor $-x_{|X|+m+1}$ can be set to $x_{|X|+m+1}$ since $x_{|X|+i} = x_{|X|+m+1} \ne 0$ for $1 \le i \le m$. Thus we can assume that all equations of types (3.18)–(3.19) are satisfied, that $x_j = \pm x_{|X|+m+1}$ for $1 \le j \le |X|$ and therefore that s equations of types (3.12)–(3.13) are fulfilled. Now $x_{|X|+m+1}$ is either positive or negative. If $x_{|X|+m+1} > 0$ it is equivalent to satisfy $10m+s$ equations of the above system and to satisfy $2m+s$ equations of the system (3.8)–(3.11). If $x_{|X|+m+1} < 0$ the truth assignment given by

$$y_j = \begin{cases} \text{TRUE} & \text{if } x_j = -x_{|X|+m+1} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

fulfills at least s clauses of the MAX 2SAT instance.

A similar construction can be used to show that MAX FLS$^\ge$ is APX-hard. For each clause C_i, $1 \le i \le m$, containing two variables x_{j_1} and x_{j_2} we consider the following equations:

$$\begin{aligned}
a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} &\ge -1 && (3.20)\\
x_{j_1},\ x_{j_2} &\ge 1 && (3.21)\\
-x_{j_1},\ -x_{j_2} &\ge -1 && (3.22)\\
x_{j_1},\ x_{j_2} &\ge -1 && (3.23)\\
-x_{j_1}, -x_{j_2} &\ge 1 && (3.24)
\end{aligned}$$

where $a_{ij} = 1$ if x_j occurs positively in C_i and $a_{ij} = -1$ if x_j occurs negatively. The overall system has $9m$ inequalities.

Clearly, any solution $\mathbf{x}$ satisfies at least two and at most three of equations (3.21)–(3.24) for each variable and when three of them are simultaneously satisfied then either $x_{j_1} = 1$ or $x_{j_1} = -1$. If in any MAX FLS$^{\geq}$ solution a variable is neither 1 nor -1, we can modify it without decreasing the number of satisfied equations. Thus we have a correspondence between solutions of the MAX 2SAT instance satisfying s clauses and solutions of the MAX FLS$^{\geq}$ instance fulfilling $6m + s$ inequalities. Moreover, opt[MAX FLS$^{\geq}$] $\leq 14 \cdot$ opt[MAX 2SAT] since opt[MAX FLS$^{\geq}$] $\leq 7m$.

As for MAX FLS$^{=}$, this L-reduction can be extended to homogeneous systems with discrete coefficients in $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ and with no pairs of identical relations. □

The question of whether this result holds for homogeneous systems with bipolar coefficients is still open. The duplication technique described in corollary 3.1.2 leads to a polynomial time reduction but not to an L-reduction because the new systems have $O(m^2)$ equations, where m is the number of clauses in the MAX 2SAT instance.

Corollary 3.2.2 MAX FLS$^{>}$ *restricted to homogeneous systems with integer coefficients is* APX*-hard.*

PROOF We adapt the L-reduction for MAX FLS$^{\geq}$ to the case with strict inequalities. For each clause C_i, $1 \leq i \leq m$, containing two variables x_{j_1} and x_{j_2} we consider the following equations:

$$a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} > -0.1 \tag{3.25}$$

$$x_{j_1},\, x_{j_2} > 0.9 \tag{3.26}$$

$$-x_{j_1},\, -x_{j_2} > -1.1 \tag{3.27}$$

$$x_{j_1},\, x_{j_2} > -1.1 \tag{3.28}$$

$$-x_{j_1},\, -x_{j_2} > 0.9 \tag{3.29}$$

where the coefficients a_{ij} are defined as above. The overall system contains $9m$ inequalities. Since we always gain by setting the variables to exactly 1 or -1, there is a correspondence between solutions of the MAX 2SAT instance satisfying s clauses and feasible solutions $\mathbf{x}$ fulfilling $6m + s$ inequalities of the above homogeneous system.

To deal with homogeneous MAX FLS$^{>}$ we add a new variable $x_{|X|+1}$ as well as new inequalities forcing $x_{|X|+1} > 0$. For each clause C_i, $1 \leq i \leq m$, we construct the following inequalities:

$$\begin{aligned}
a_{ij_1}x_{j_1} + a_{ij_2}x_{j_2} + 0.1x_{|X|+1} &> 0 && (3.30)\\
x_{j_1} - 0.9x_{|X|+1} &> 0 && (3.31)\\
-x_{j_1} + 1.1x_{|X|+1} &> 0 && (3.32)\\
x_{j_1} + 1.1x_{|X|+1} &> 0 && (3.33)\\
-x_{j_1} - 0.9x_{|X|+1} &> 0 && (3.34)\\
x_{j_2} - 0.9x_{|X|+1} &> 0 && (3.35)\\
-x_{j_2} + 1.1x_{|X|+1} &> 0 && (3.36)\\
x_{j_2} + 1.1x_{|X|+1} &> 0 && (3.37)\\
-x_{j_2} - 0.9x_{|X|+1} &> 0 && (3.38)\\
x_{|X|+1} &> 0 && (3.39)
\end{aligned}$$

with the usual coefficients. Thus we have a system with $10m$ strict inequalities. Here the correspondence is between solutions of the MAX 2SAT instance satisfying s clauses and solutions fulfilling $7m + s$ inequalities. It is easy to see that any solution $\mathbf{x}$ which satisfies $7m + s$ inequalities must have $x_{|X|+1} > 0$. In fact, if $x_{|X|+1} = 0$ then $\mathbf{x}$ can satisfy at most $5m$ inequalities and if $x_{|X|+1} < 0$ it can satisfy at most $7m$ inequalities of types (3.30)–(3.38).

In order to avoid having pairs of identical relations we just multiply the identical inequalities by different integer factors. □

Theorem 3.2.1 and corollary 3.2.2 are valid for homogeneous systems with integer coefficients and no identical relations. If identical relations are allowed, theorem 3.2.1 holds for systems with ternary coefficients while corollary 3.2.2 for systems with bipolar ones. The L-reduction from MAX 2SAT to MAX FLS$^{>}$ can be extended to bipolar coefficients by proceeding like in the proof of theorem 3.1.4. We multiply every strict inequality by a factor 10 and we apply Carver's result as well as the duplication technique. Note that when the new (equivalent) systems are constructed the number of equations is just multiplied by a constant factor.

The following results give a better characterization of the approximability of MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ in terms of the various classes mentioned in chapter 2.

Homogeneous MAX FLS$^=$ can obviously be approximated within $p/\min\{n-1, p\}$, where p is the number of equations and n the number of variables occurring in the system. The next proposition shows that it cannot be approximated within any constant factor.

Proposition 3.2.3 MAX FLS$^=$ *restricted to homogeneous systems with integer coefficients is not in* APX *unless* P = NP.

PROOF Suppose that MAX FLS$^=$ can be approximated within a constant $c > 1$ and consider an arbitrary instance with p homogeneous equations $e_1 = 0, \ldots, e_p = 0$. Let s be the number of equations contained in a maximum feasible subsystem.

Construct a new problem with the equations $e_{i,j,k} = 0$ where $e_{i,j,k} = e_i + k \cdot e_j$, $1 \leq i \leq p$, $1 \leq j \leq p$, $1 \leq k \leq T$ for an integer T. Since $e_i = 0$ and $e_j = 0$ imply $e_{i,j,k} = 0$ for every value of k, the s satisfied equations of the original problem give $T \cdot s^2$ satisfied equations of the new problem. However, some additional equations may be satisfied when $e_i = -k \cdot e_j$ and $e_i \neq 0$. But no more than p^2 equations are fulfilled in such a way because there is at most one such equation for each pair (i, j).

Since the optimal solution contains at least $T \cdot s^2$ satisfied equations, the approximation algorithm provides a solution that fulfills at least $T \cdot s^2/c$ equations. We examine the satisfied equations and throw away every equation $e_i + k \cdot e_j$ where $e_i \neq 0$. This leaves us with at least $T \cdot s^2/c - p^2$ equations. Since there are at most T equations for every pair (i, j), we obtain at least

$$\sqrt{\frac{T \cdot s^2/c - p^2}{T}} = \sqrt{\frac{s^2}{c} - \frac{p^2}{T}}$$

satisfied equations of the original problem. If we run the approximation algorithm directly on the original problem we are guaranteed to find s/c satisfied equations.

By choosing

$$T \geq \left\lceil \frac{p^2 c^2}{s^2(c-1)} \right\rceil + 1,$$

more equations are satisfied by applying the approximation algorithm to the $e_{i,j,k}$ problem than by applying it to the original problem. This can be done over and over again to get better constants in the approximation. But theorem 3.2.1 states that MAX FLS$^=$ is APX-hard and thus there exists a constant δ, with $0 < \delta < 1$, such that it cannot be approximated within a smaller constant than $1/(1-\delta)$. Hence MAX FLS$^=$ is not in APX. $\square$

By using tuples of $\log p$ equations instead of pairs of equations and by using walks on expander graphs in order to choose a polynomial number of these tuples it is possible to show a yet stronger result.

Theorem 3.2.4 *Assuming* P $\neq$ NP*, there is a positive constant ε such that homogeneous* MAX FLS$^=$ *cannot be approximated within p^ε, where p is the number of equations.*

PROOF We start as in the proof of proposition 3.2.3 with an arbitrary instance of MAX FLS$^=$ with p homogeneous equations $e_1 = 0, \ldots, e_p = 0$ and let s be the number of equations contained in a maximum feasible subsystem. In this proof we assume that a fixed percentage of the equations can be satisfied, i.e. that $s/p \geq \beta$ where β is a constant. The APX-hard problem instances constructed in the first reduction of theorem 3.2.1 have $\beta > 10/14$, but we can increase β by adding a lot of trivially satisfiable equations in a new variable.

Instead of using pairs of equations when constructing the new problem we take m-tuples of equations, where m is about $\log p$. Let us consider

$$e_{i,k} = \sum_{j=1}^{m} e_{i_j} k^j \quad \text{with} \quad 1 \leq i \leq N, 1 \leq k \leq T,$$

for some integers N and T to be determined. For each i with not all $e_{i_j} = 0$ the polynomial equation $\sum e_{i_j} x^j = 0$ can have at most m solutions of which one is $x = 0$. Thus at most $m - 1$ of the T equations $e_{i,k} = 0, 1 \leq k \leq T$, can be satisfied unless $e_{i_j} = 0$ for all j in $[1..m]$. An m-tuple for which every $e_{i_j} = 0$ is called a *good tuple*.

The problem here is that we cannot form new equations from every m-tuple of old equations because there would be an exponential number of them, namely $p^m \sim p^{\log p}$. We would like to select a polynomial subset of all possible m-tuples such that the proportion of good tuples in this subset is about the same as the proportion of good tuples in the set of all possible m-tuples, which is s^m/p^m.

Therefore we need the following lemma, proved by Alon using [5, 98].

Lemma 3.2.5 [4] *Let G be a d-regular graph with p nodes in which the absolute value of every nontrivial eigenvalue of the adjacency matrix is at most λd. Let S be a set of s nodes of G and $\mu = s/p$. Let $P = P(S, m)$ be the total number of walks of m nodes that stay in S (i.e. the number of paths of length m with every node in S). Then for every odd integer m we have*

$$\mu\left(\mu - \lambda(1-\mu)\right)^{m-1} \leq \frac{P}{p \cdot d^{m-1}} \leq \mu\left(\mu + \lambda(1-\mu)\right)^{m-1}.$$

We will consider Ramanujan graphs as G. By definition, a Ramanujan graph is a connected d-regular graph whose eigenvalues are either $\pm d$ or at most $2\sqrt{d-1}$ in absolute value. Thus we can take $\lambda = 2/\sqrt{d}$. As proved in [98, 119], infinite families of Ramanujan graphs can be constructed for each d such that $d-1$ is a prime congruent to 1 modulo 4.

Choose m as the least odd integer greater than $\log p$ (where log is the logarithm in base 2). We associate each node in G to an equation $e_i = 0$, $1 \le i \le p$. As m-tuples in the constructed problem we use every possible path of length m in G. Since there are $p \cdot d^{m-1}$ such paths in G, $N = p \cdot d^{m-1}$.

If S is the set of nodes corresponding to the s equations contained in some maximum feasible subsystem, $P(S, m)$ will be exactly the number of good m-tuples.

By assigning variables as in the optimal solution of the original problem we will satisfy at least $T \cdot P(S, m)$ equations, so we know that any optimal solution of the new problem fulfills at least

$$NT\frac{s}{p}\left(\frac{s}{p} - \lambda\left(1 - \frac{s}{p}\right)\right)^{m-1} \tag{3.40}$$

(rounded to an integer) equations. On the other hand, we know that if the optimal solution of the original problem only had satisfied $\lfloor s(1-\delta)\rfloor$ equations, for some δ with $0 < \delta < 1$, then any optimal solution of the new problem would fulfill at most

$$NT\frac{s(1-\delta)}{p}\left(\frac{s(1-\delta)}{p} + \lambda\left(1 - \frac{s(1-\delta)}{p}\right)\right)^{m-1} + N(m-1) \tag{3.41}$$

(rounded to an integer) equations.

We will show that the quotient between (3.40) and (3.41) can be bounded from below by $(NT)^{\varepsilon}$ for some $\varepsilon > 0$. If there exists an algorithm approximating MAX FLS$^=$ within p^{ε} (where p is the number of equations) we can apply it to the constructed problem with NT equations and be sure that it provides a solution containing more satisfied equations than (3.41). Therefore the assignment given by the algorithm will approximate the original problem within $1/(1-\delta)$, but this is NP-hard since MAX FLS$^=$ is APX-hard according to theorem 3.2.1.

Thus we just have to bound the quotient between (3.40) and (3.41) appropriately.

Let $\lambda = \alpha\beta$ where α is a positive constant to be defined later. Then $\alpha\beta = 2/\sqrt{d}$ and $\log d = 2 + 2\log\frac{1}{\alpha\beta}$. Hence $N = p \cdot d^{m-1} \approx p \cdot p^{\log d} = p^{3+2log\frac{1}{\alpha\beta}}$ and

$$(3.40) > NT\frac{s}{p}\left(\frac{s}{p} - \alpha\beta\right)^{m-1} \ge NT\frac{s}{p}\left(\frac{s}{p}(1-\alpha)\right)^{m-1} \tag{3.42}$$

$$\ge NT\beta\left(\beta(1-\alpha)\right)^{m-1} \approx NT\beta p^{-\log\frac{1}{\beta(1-\alpha)}} \tag{3.43}$$

First we consider the first term of (3.41):

$$NT\frac{s(1-\delta)}{p}\left(\frac{s(1-\delta)}{p}+\lambda\left(1-\frac{s(1-\delta)}{p}\right)\right)^{m-1} \le NT\frac{s(1-\delta)}{p}\left(\frac{s(1-\delta)}{p}+\lambda\right)^{m-1}$$

$$= NT\frac{s(1-\delta)}{p}\left(\frac{s(1-\delta)}{p}+\alpha\beta\right)^{m-1} \le NT\frac{s(1-\delta)}{p}\left(\frac{s}{p}(1-\delta+\alpha)\right)^{m-1} \tag{3.44}$$

Using (3.42) and (3.44) we can bound the quotient between (3.40) and the first term of (3.41) from below by

$$\frac{NT\frac{s}{p}\left(\frac{s}{p}(1-\alpha)\right)^{m-1}}{NT\frac{s(1-\delta)}{p}\left(\frac{s}{p}(1-\delta+\alpha)\right)^{m-1}} = \frac{1}{1-\delta}\left(\frac{1-\alpha}{1-\delta+\alpha}\right)^{m-1} \approx \frac{1}{1-\delta}p^{\log\frac{1-\alpha}{1-\delta+\alpha}} > p^{\log\frac{1-\alpha}{1-\delta+\alpha}}. \tag{3.45}$$

We now consider $N(m-1)$, the second term of (3.41). In this case, we have

$$\frac{(3.40)}{N(m-1)} \ge \frac{NT\beta p^{-\log\frac{1}{\beta(1-\alpha)}}}{N\log p} \ge Tp^{-\log\frac{1}{\beta(1-\alpha)}-\delta'} \text{ for every } \delta' > 0.$$

We would like both $p^{\log\frac{1-\alpha}{1-\delta+\alpha}}$ and $Tp^{-\log\frac{1}{\beta(1-\alpha)}-\delta'}$ to be greater than $2(NT)^\varepsilon$ for some constant $\varepsilon > 0$. If we choose $T = \left\lceil p^{\log\frac{1}{\beta(1-\delta+\alpha)}}\right\rceil$ the second expression is greater than

$$p^{\log\frac{1}{\beta(1-\delta+\alpha)}-\log\frac{1}{\beta(1-\alpha)}-\delta'} = p^{\log\frac{1-\alpha}{1-\delta+\alpha}-\delta'}$$

which is less than the first quotient $p^{\log\frac{1-\alpha}{1-\delta+\alpha}}$ for every $\delta' > 0$. Thus we only have to bound $p^{\log\frac{1-\alpha}{1-\delta+\alpha}-\delta'}$ from below by

$$2(NT)^\varepsilon \approx p^{\varepsilon\left(3+2\log\frac{1}{\alpha\beta}+\log\frac{1}{\beta(1-\delta+\alpha)}\right)} = p^{\varepsilon\left(3+3\log\frac{1}{\beta}+2\log\frac{1}{\alpha}+\log\frac{1}{1-\delta+\alpha}\right)}.$$

Let $\alpha = \varepsilon'\delta$ for some $\varepsilon' > 0$. Then we must satisfy

$$\varepsilon < \frac{\log\frac{1-\varepsilon'\delta}{1-\delta+\varepsilon'\delta}-\delta'}{3+3\log\frac{1}{\beta}+2\log\frac{1}{\varepsilon'\delta}+\log\frac{1}{1-\delta+\varepsilon'\delta}} \approx \frac{\delta(1-2\varepsilon')-\delta'\ln 2}{3\ln 2+3\ln\frac{1}{\beta}+2\ln\frac{1}{\varepsilon'}+2\ln\frac{1}{\delta}+\delta(1-\varepsilon')}$$

which, given β and choosing δ' and ε' small enough, is a positive constant, slightly smaller than $\delta/\ln(1/\delta^2)$. □

Note that MAX FLS$^=$ can trivially be approximated within p by satisfying a single equation.

MAX FLS$^\ge$ and MAX FLS$^>$ turn out to be much easier to approximate than MAX FLS$^=$.

Proposition 3.2.6 MAX FLS$^\mathcal{R}$ *with* $\mathcal{R} \in \{\ge, >\}$ *is* APX-*complete and can be approximated within* 2.

PROOF For both problems we use the following simple algorithm.

Algorithm:

```
Input: An instance (A,b) of Max FLS^R with R ∈ {≥,>}
Init:  X := {variables occurring in (A,b)} and E := {inequalities in (A,b)}
 WHILE E ≠ ∅ DO
 IF there are inequalities in E that contain a single variable THEN
    U := {x ∈ X | x occurs as a single variable in at least one ineq.  of E}
    Pick at random y ∈ U
    F(y) := {e ∈ E | e contains only the variable y}
    Assign a value to y satisfying as many ineq.  in F(y) as possible
    E := E − F(y)
 ELSE
    Pick at random a variable y and assign a random value to it
    Reevaluate the inequalities in E that contain y
 END IF
 X := X − {y}
 END WHILE
```

This algorithm is guaranteed to provide a 2-approximation because we can always assign to y a value which satisfies at least half of the inequalities in $F(y)$. Moreover, it runs in polynomial time since each variable and each inequality are considered only once.

Since Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{\geq, >\}$ is Apx-hard and can be approximated within 2, both problems are Apx-complete. □

Notice that this greedy-like method is similar to the 2-approximation algorithm which has been proposed for Max Sat [93]. As for Max Sat and Max Cut, there could exist a better polynomial time algorithm that guarantees a smaller performance ratio [167, 72].

Provided that P $\neq$ NP, the previous results describe the approximability of Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ quite well. While Max FLS$^{=}$ cannot be approximated within p^{ε} for some $\varepsilon > 0$ where p is the number of equations, Max FLS$^{\geq}$ and Max FLS$^{>}$ can be approximated within a factor 2 but not within every constant.

One can observe that Max FLS$^{\geq}$ and Max FLS$^{>}$ share a common property: a constant fraction of the relations can always be simultaneously satisfied. The above 2-approximation algorithm is optimal in the sense that no constant fraction larger than 1/2 can be guaranteed. This property does not hold for Max FLS$^{=}$ since in the system

$x_1 = k$ with $k \in [1..p]$ at most one equation can be satisfied, that is only $1/p$ of the total number of relations.

3.2.2 Max FLS$^=$ in finite fields: a particular case

We now consider an interesting special case of Max FLS$^=$ which admits a polynomial time approximation algorithm. The problem, named Max FLS$^=$ over $GF(q)$, naturally arises when the equations of the system are restricted to be over a finite field $GF(q)$, that is modulo a prime q.

Proposition 3.2.7 *For any prime q,* Max FLS$^=$ *over $GF(q)$ is* Apx*-complete and approximable within q.*

Proof The L-reduction from Max 2Sat to Max FLS$^=$ can be used to show that Max FLS$^=$ over $GF(q)$ is Apx-hard for $q \geq 3$. However, it breaks down for $q = 2$ because in that case an equation of the type $x_i + x_j = 0$ has both solutions $x_i = x_j = 0$ and $x_i = x_j = 1$.

For $q = 2$ we proceed by reduction from Max Cut where, given a graph $G = (V, E)$, one looks for a set of nodes $V' \subseteq V$ such that the number of edges $[v_i, v_j] \in E$ with $v_i \in V'$ and $v_j \in V - V'$ is as large as possible.

For every edge $[v_i, v_j]$ we introduce the equation $x_i + x_j = 1 \pmod 2$. An equation is satisfied if and only if one of the variables is odd and the other is even. The oddness and evenness partition the graph and the size of the cut is exactly the number of satisfied equations. This construction is clearly an L-reduction.

When the equations are in $GF(q)$, it is easy to find a solution which satisfies at least $1/q$ of the equations. This can be achieved using a simple greedy algorithm similar to that presented in the proof of proposition 3.2.6. □

For example, if all coefficients take their values in $\{0, 1\}$ and if all computations are performed modulo 2, Max FLS$^=$ can be approximated within 2. This result is clearly not applicable when standard computations are used.

Using proposition 3.2.7 and proof techniques from theorem 3.2.4 we can show a better lower bound on the approximability of Max FLS$^=$ over $GF(q)$.

Theorem 3.2.8 *For any prime q there is a constant $\varepsilon > 0$ such that* Max FLS$^=$ *over $GF(q)$ cannot be approximated within q^ε.*

PROOF We use the same construction as in the proof of theorem 3.2.4 but we choose m as the least odd number greater than $\log q$ (instead of $\log p$). Then $N = p \cdot d^{m-1} \approx p \cdot q^{\log d}$. We let $T = q - 1$, which means that we consider every number in $GF(q)$ except 0 as constants in the constructed equations.

Since MAX FLS$^=$ over $GF(q)$ is APX-hard there is a constant $\delta > 0$ such that it cannot be approximated within $1/(1-\delta)$. The quotient between (3.40) and the second term of (3.41) now is

$$\frac{(3.40)}{N(m-1)} \geq \frac{NT\beta q^{-\log\frac{1}{\beta(1-\alpha)}}}{N\log q} \geq q^{1-\log\frac{1}{\beta(1-\alpha)}-\delta'} \text{ for every } \delta' > 0,$$

and the quotient between (3.40) and the first term of (3.41) is, using (3.43) and (3.44), bounded by

$$\frac{NT\frac{s}{p}\left(\frac{s}{p}(1-\alpha)\right)^{m-1}}{NT\frac{s(1-\delta)}{p}\left(\frac{s}{p}(1-\delta+\alpha)\right)^{m-1}} \approx \frac{1}{1-\delta} q^{\log\frac{1-\alpha}{1-\delta+\alpha}} \geq q^{\log\frac{1-\alpha}{1-\delta+\alpha}}.$$

Thus both quotients are bounded by q^{ε} for some $\varepsilon > 0$. $\square$

Thus when the equations are restricted to be over a finite field modulo a prime q we have the same type of nonapproximability bound as for the general problem MAX FLS$^=$ but with respect to q instead of p.

3.2.3 Weighted and constrained MAX FLS

As mentioned in chapter 1, in many practical situations different relations may have different importances. This gives rise to the weighted versions of MAX FLS where a weight is assigned to each relation and one looks for a solution that maximizes the total weight of the satisfied relations.

Proposition 3.2.9 *Weighted* MAX FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *and positive integer weights bounded by a polynomial in the input size are as hard to approximate as the corresponding unweighted versions.*

PROOF Basic MAX FLS$^{\mathcal{R}}$ is clearly a special case of the weighted variant where all weights are equal to one. In order to prove the other direction, it suffices to note that any weighted instance can be associated to an equivalent unweighted one. If the weights are polynomially bounded integers, we just need to make for each relation a number of copies equal to the corresponding weight. The resulting unweighted instance has clearly a polynomial number of relations. □

Interesting and important special cases of weighted Max FLS include the constrained versions, denoted by C Max FLS, where some relations are mandatory while the others are optional. The objective is then to find a solution that satisfies all mandatory relations and as many optional ones as possible. Any instance of C Max FLS is equivalent to the particular instance of weighted Max FLS where each optional relation is assigned a unit weight and each mandatory relation a weight larger than the total number of optional ones.

However, while the polynomially bounded weighted versions of Max FLS are as hard to approximate as the unweighted versions, most of the constrained versions turn out to be at least as hard as Max Ind Set. Thus, unless P = NP, they cannot be approximated within m^ε for some $\varepsilon > 0$, where m is the size of the instance.

When considering mixed variants of C Max FLS with different types of mandatory and optional relations, C Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ with $\mathcal{R}_1$, $\mathcal{R}_2 \in \{=, \geq, >\}$ denotes the variant where the mandatory relations are of type $\mathcal{R}_1$ and the optional ones of type $\mathcal{R}_2$.

Theorem 3.2.10 C Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ *with* $\mathcal{R}_1$*,* $\mathcal{R}_2 \in \{\geq, >\}$ *is* Max Ind Set*-hard even for homogeneous systems with ternary coefficients.*

Proof The proof is by cost preserving polynomial time transformations from Max Ind Set. We proceed like in the first part of the reduction of theorem 3.1.4 and start with C Max FLS$^{>;>}$.

Let $G = (V, E)$ be an arbitrary instance of Max Ind Set. For each edge $[v_i, v_j] \in E$ we construct the mandatory inequality

$$x_i + x_j < 0 \tag{3.46}$$

and for each $v_i \in V$ the optional inequality

$$x_i > 0. \tag{3.47}$$

Thus we have a system with $|V|$ variables and $|E| + |V|$ strict inequalities. As shown in the proof of theorem 3.1.4, the given graph G contains an independent set I of size s if and only if there exists a solution $\mathbf{x}$ satisfying all mandatory inequalities and s optional ones.

This cost preserving transformation can easily be adapted to show that the other problems are MAX IND SET-hard. For C MAX FLS$^{\geq;>}$ we just change the mandatory inequality of type (3.46) to $x_i + x_j \leq 0$.

For C MAX FLS$^{\geq;\geq}$ an additional variable x_{n+1} needed. We include the mandatory inequality

$$x_{n+1} \geq 0 \tag{3.48}$$

and for each edge $[v_i, v_j] \in E$ the mandatory inequality

$$x_i + x_j \leq 0. \tag{3.49}$$

For each $v_i \in V$ we consider the optional inequality

$$x_i - x_{n+1} \geq 0. \tag{3.50}$$

Thus we have a system with $|V| + 1$ variables and $|E| + |V| + 1$ inequalities.

Now the given graph G contains an independent set I of size s if and only if there exists a solution $\mathbf{x}$ satisfying all inequalities of types (3.48)–(3.49) and s inequalities of type (3.50). Given an independent set $I \subseteq V$ of size s, the solution obtained by setting

$$x_i = \begin{cases} 1 & \text{if } v_i \in I \text{ or } i = n+1 \\ -2 & \text{if } 1 \leq i \leq |V| \text{ and } v_i \notin I \end{cases}$$

satisfies all mandatory inequalities and all the optional ones that correspond to a node $v_i \in I$. The $|V| - s$ optional inequalities associated with $v_i \notin I$ are not fulfilled because $x_i < x_{n+1}$. Conversely, given an appropriate solution $\mathbf{x}$ that fulfills all mandatory inequalities and s optional ones. Since the variable x_{n+1} is included in every optional inequality and hence is the most common one, it cannot be zero and must be positive because of relation (3.48). The set $I \subseteq V$ containing all nodes associated to variables with values larger than x_{n+1} is then an independent set of cardinality s.

Finally, by simply changing the mandatory inequalities in this reduction to strict inequalities we get a reduction to C MAX FLS$^{>;\geq}$. □

Thus forcing a subset of relations makes MAX FLS$^{\mathcal{R}}$ harder for $\mathcal{R} \in \{\geq, >\}$: MAX FLS$^{\mathcal{R}}$ is APX-complete while C MAX FLS$^{\mathcal{R};\mathcal{R}}$ is MAX IND SET-hard. This is not true for MAX FLS$^{=}$ since any instance of C MAX FLS$^{=;\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ can be transformed into an equivalent instance of MAX FLS$^{\mathcal{R}}$ by eliminating variables in the set of optional equations using the set of mandatory ones.

Nevertheless two simple variants of C MAX $\text{FLS}^{=;=}$ and MAX $\text{FLS}^{=}$ turn out to be MAX IND SET-hard. Replacing the strict inequality of type (3.46) by $x_i + x_j \leq 1$ and that of type (3.47) by $x_i = 1$, one verifies that C MAX $\text{FLS}^{\geq;=}$ is MAX IND SET-hard even for systems with ternary coefficients and bipolar right hand side components. This has an immediate implication on the hardness of MAX $\text{FLS}^{=}$ with the natural nonnegativeness constraint that arises, for instance, when dealing with infeasible linear programs.

Corollary 3.2.11 MAX $\text{FLS}^{=}$ *restricted to systems with ternary coefficients and nonnegative variables is* MAX IND SET*-hard.*

PROOF By adding a nonnegative slack variable for each mandatory inequality of any C MAX $\text{FLS}^{\geq;=}$ instance, we obtain a particular instance of C MAX $\text{FLS}^{=;=}$ that can be transformed into an equivalent instance of MAX $\text{FLS}^{=}$. Any variable x_i unrestricted in sign can then be replaced by two nonnegative variables x_i^1 and x_i^2. The coefficients of these two auxiliary variables have the same absolute value as the coefficient of x_i but opposite signs. □

The question of whether MAX $\text{FLS}^{=}$ becomes harder to approximate when the variables are restricted to be nonnegative or whether the basic version is already MAX IND SET-hard, is still open. Note that the positiveness constraints do not affect C MAX $\text{FLS}^{\geq;\geq}$ since they can be viewed as additional mandatory inequalities.

The approximability of mixed variants involving $\neq$ mandatory relations is somewhat different.

Proposition 3.2.12 *While* C MAX $\text{FLS}^{\neq;=}$ *is* MAX IND SET*-hard,* C MAX $\text{FLS}^{\neq;>}$ *and* C MAX $\text{FLS}^{\neq;\geq}$ *are* APX*-complete and can be approximated within 2.*

PROOF For C MAX $\text{FLS}^{\neq;=}$, we proceed by cost preserving transformation from MAX IND SET. For each edge $[v_i, v_j] \in E$ we consider the mandatory relation $x_i + x_j \neq 2$ and for each node $v_i \in V$ we consider the optional relation $x_i = 1$. Clearly, there exists an independent set of size s if and only if there exists a solution $\mathbf{x}$ satisfying all mandatory relations and s optional ones.

According to theorem 3.2.1 and corollary 3.2.2, MAX $\text{FLS}^{\geq}$ and MAX $\text{FLS}^{>}$ are APX-hard and the constrained versions C MAX $\text{FLS}^{\neq;\geq}$ and C MAX $\text{FLS}^{\neq;>}$ must be at least as hard. To approximate these problems within 2 we modify the greedy algorithm in proposition 3.2.6 so that it also takes into account the mandatory relations. When a variable value is chosen it should not contradict any of the mandatory relations that have a single unassigned variable. This is always possible since there is only a finite number of such relations while the number of possible values satisfying the largest number of optional relations is infinite. □

3.2.4 Max FLS with bounded discrete variables

In this section we derive strong lower bounds on the approximability of Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ when the variables are restricted to take a finite number of discrete values. Both extreme cases with binary variables in $\{0, 1\}$ and bipolar variables in $\{-1, 1\}$ are considered. The corresponding variants of Max FLS are named Bin Max FLS$^{\mathcal{R}}$ and Bip Max FLS$^{\mathcal{R}}$ respectively.

Theorem 3.2.13 Bin Max FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is* Max Ind Set*-hard even for systems with ternary coefficients.*

Proof The proof is of course by cost preserving transformation from Max Ind Set. We first consider Max FLS$^{>}$.

Let $G = (V, E)$ be the graph of an arbitrary instance of Max Ind Set. For each node $v_i \in V$ we construct the strict inequality

$$x_i - \sum_{j \in N(v_i)} x_j > 0$$

where j belongs to $N(v_i)$ if and only if v_j is adjacent to v_i. Thus we have a system of $|V|$ homogeneous inequalities with ternary coefficients. By construction, the ith inequality is satisfied if and only if $x_i = 1$ and $x_j = 0$ for all j, $1 \leq j \leq |V|$.

It is easy to verify that given an independent set $I \subseteq V$ of size s we get a binary solution satisfying the s corresponding inequalities by setting $x_i = 1$ if $v_i \in I$ and $x_i = 0$ otherwise. Conversely, given any binary solution $\mathbf{x}$ satisfying s inequalities we obtain an independent set of size s by including in I all nodes v_i, $1 \leq i \leq |V|$, such that $x_i = 1$.

Notice that this cost preserving polynomial transformation works also for $A\mathbf{x} \geq \mathbf{1}$ or $A\mathbf{x} = \mathbf{1}$. This construction can be adapted to show that Bin Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ is hard to approximate even when restricted to homogeneous systems. However, we must allow the coefficients to take their values in $\{-2, 0, 1\}$ instead of in $\{-1, 0, 1\}$. □

Corollary 3.2.14 BIP MAX FLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is* MAX IND SET*-hard even for systems with ternary coefficients and integer right hand side components.*

PROOF By simple cost preserving transformation from BIN MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$. For any relation

$$\sum_{j=1}^{n} a_{ij} x_j \ \mathcal{R} \ b_i$$

with binary variables $x_j \in \{0,1\}$ and $1 \leq i \leq p$, we can construct an equivalent relation

$$\sum_{j=1}^{n} a_{ij} y_j \ \mathcal{R} \ 2b_i + \sum_{j=1}^{n} a_{ij}$$

with bipolar variables $y_j \in \{-1, 1\}$ using the variable substitution $y_j = 2x_j - 1$. □

Although the above transformation does not preserve homogeneity, we know from the L-reductions used to prove theorem 3.2.1 and corollary 3.2.2 that homogeneous BIP MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ is APX-hard. In fact, homogeneous BIP MAX FLS$^{\geq}$ and BIP MAX FLS$^{>}$ belong to APX.

Proposition 3.2.15 *Homogeneous* BIP MAX FLS$^{\geq}$ *can be approximated within 2 and homogeneous* BIP MAX FLS$^{>}$ *can be approximated within 4.*

PROOF We first deal with homogeneous BIP MAX FLS$^{\geq}$. Take an arbitrary bipolar vector $\mathbf{x}$ and consider the number of satisfied relations for $\mathbf{x}$ and $-\mathbf{x}$. If the left hand side of a relation is positive for $\mathbf{x}$ it will be negative for $-\mathbf{x}$ and vice versa. Thus one of these antipodal vectors satisfies at least half of the inequalities.

This trivial algorithm does not work for homogeneous BIP MAX FLS$^{>}$ because many relations may be zero for both antipodal vectors. Therefore we first look for a solution with many nonzero relations. A greedy approximation algorithm similar to the one in proposition 3.2.6 provides a solution $\mathbf{x}$ for which at least half of the relations are nonzero. Now one of $\mathbf{x}$ and $-\mathbf{x}$ makes at least half of these relations, and therefore a quarter of all relations, positive. □

Thus restricting the systems to be homogeneous makes BIP MAX FLS$^{\geq}$ and BIP MAX FLS$^{>}$ much easier to approximate. The situation is quite different for homogeneous BIP MAX FLS$^{=}$ with integer coefficients. According to the same arguments as in the proof of theorem 3.2.4, this problem cannot be approximated within p^{ε} for some $\varepsilon > 0$.

Proposition 3.2.16 BIN MAX FLS$^{\neq}$ *and* BIP MAX FLS$^{\neq}$ *are* APX*-complete and can be approximated within* 2*.*

PROOF For both problems the proof is by cost preserving transformation from MAX 2SAT. We first consider BIP MAX FLS$^{\neq}$. Let $(X, \mathcal{C})$ be an arbitrary instance of MAX 2SAT with $\mathcal{C} = \{C_1, \ldots, C_m\}$. For each clause C_i, $1 \leq i \leq m$, containing two variables x_{j_1} and x_{j_2} we consider the relation

$$a_{j_1} x_{j_1} + a_{j_2} x_{j_2} \neq -2$$

where $a_j = 1$ if the Boolean variable x_j occurs positively in C_i, $a_j = -1$ if x_j occurs negatively in C_i, and $a_j = 0$ otherwise. Thus we have a system with m relations. Clearly, there exists a truth assignment satisfying s clauses of $(X, \mathcal{C})$ if and only if there exists a solution $\mathbf{x}$ satisfying s relations.

A similar reduction is used for BIN MAX FLS$^{\neq}$. For each clause of the MAX 2SAT instance, we construct a relation according to the following table:

Clause	Relation
$x_{j_1} \vee x_{j_2}$	$x_{j_1} + x_{j_2} \neq 0$
$\bar{x}_{j_1} \vee x_{j_2}$	$-x_{j_1} + x_{j_2} \neq -1$
$\bar{x}_{j_1} \vee \bar{x}_{j_2}$	$-x_{j_1} - x_{j_2} \neq -2$
x_{j_1}	$x_{j_1} \neq 0$
$\bar{x}_{j_1}$	$x_{j_1} \neq 1$

where $\bar{x}_j$ denotes the negation of x_j.

Both problems are in APX since they can be approximated within 2 using a greedy algorithm similar to the one in proposition 3.2.6. □

The following result shows that the constrained variants of BIN MAX FLS$^{\mathcal{R}}$ with mandatory relations, named C BIN MAX FLS$^{\mathcal{R}_1;\mathcal{R}_2}$, are NPO PB-complete, that is, at least as hard to approximate as every NP optimization problem with polynomially bounded objective function.

Proposition 3.2.17 C BIN MAX FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ *is* NPO PB*-complete for* $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$ *even for systems with ternary coefficients and integer right hand side components.*

PROOF We first show the result for the problem C BIN MAX FLS$^{>;>}$ and then extend the result to the other variants.

We proceed by cost preserving transformations from Max Dones that is known to be NPO PB-complete [99]. Given two disjoint sets X, Z of variables and a collection $\mathcal{C} = \{C_1, \dots, C_m\}$ of disjunctive clauses of at most 3 literals, find a truth assignment for X and Z that satisfies every clause in $\mathcal{C}$ so that the number of Z variables that are set to TRUE in the assignment is maximized.

Consider an arbitrary instance of Max Dones with the variables $y_1, \dots, y_n$ where $y_j \in Z$ if $1 \leq j \leq |Z|$ and $y_j \in X$ if $|Z| < j \leq n$. For each clause $l_{j_1} \vee l_{j_2} \vee l_{j_3} \in \mathcal{C}$ we construct the mandatory inequality

$$t_{j_1} + t_{j_2} + t_{j_3} > 0 \tag{3.51}$$

where, for $1 \leq k \leq 3$, $t_{j_k} = x_j$ if $l_{j_k} = y_j$, $t_{j_k} = 1 - x_j$ if $l_{j_k} = \bar{x}_j$, and $t_{j_k} = 0$ if there is no l_{j_k} (i.e. if the clause contains less than three literals). For each variable $y_j \in Z$ with $1 \leq j \leq |Z|$ we consider the optional inequality

$$x_j > 0. \tag{3.52}$$

Thus we have a system with $|X| + |Z|$ variables, $|Z|$ optional relations and m mandatory ones.

We claim that there is an assignment to X and Z with s variables from Z set to TRUE if and only if there exists a solution $\mathbf{x} \in \{0,1\}^n$ that satisfies s optional relations of the corresponding linear system. Given an assignment, the solution $\mathbf{x}$ defined by

$$x_j = \begin{cases} 1 & \text{if the Boolean variable } y_j \text{ is set to TRUE} \\ 0 & \text{otherwise} \end{cases}$$

satisfies all mandatory relations and s of the optional ones. Conversely, given a solution vector $\mathbf{x}$ that satisfies s of the optional relations, the corresponding assignment (y_j is TRUE if and only if $x_j = 1$) satisfies all clauses and s of the Z variables are TRUE.

For the other constrained problems C Bin Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$, we use the same reduction as above but the right hand side of the relations must be substituted according to Table 3.1. In the case of mandatory equations we need to introduce two additional slack variables x' and x'' in each equation (a total of $2m$ new variables). $\square$

These results imply, using the same argument as in corollary 3.2.14, that the corresponding bipolar versions C Bip Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ with $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$ are NPO PB-complete for systems with ternary coefficients and integer right hand side components.

	operator $\geq$	operator $>$	operator $\neq$	operator $=$
type (3.51)	≥ 1	> 0	$\neq 0$	$= 1 + x' + x''$
type (3.52)	≥ 1	> 0	$\neq 0$	$= 1$

Table 3.1: Right hand side values used in the C Bin Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ reductions for all the possible choices of $\mathcal{R}_1$ and $\mathcal{R}_2$. The two types of optional and mandatory relations are mentioned.

Since Max Dones cannot be approximated within $|Z|^{1-\varepsilon}$, for any $\varepsilon > 0$, unless P = NP [48], the same nonapproximability bound $\hat{p}^{1-\varepsilon}$ is valid for all versions of C Bin Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$ and C Bip Max FLS$^{\mathcal{R}_1;\mathcal{R}_2}$, where $\hat{p}$ is the number of optional relations.

3.2.5 Summary of Max FLS results

The various versions of Max FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$ that we have considered are obtained by placing constraints on the coefficients (left and right hand sides), on the variables and on the relations that must be satisfied.

	Real variables	Binary variables
Max FLS$^{=}$	not within p^{ε} for some $\varepsilon > 0$	Max Ind Set-hard
Max FLS$^{\geq}$	Apx-complete (within 2)	
Max FLS$^{\neq}$	trivial	Apx-complete (within 2)
C Max FLS$^{=;=}$	not within p^{ε} for some $\varepsilon > 0$	NPO PB-complete
C Max FLS$^{=;\geq}$	Apx-complete (within 2)	
C Max FLS$^{\geq;=}$	Max Ind Set-hard	
C Max FLS$^{\geq;\geq}$		
C Max FLS$^{\neq;=}$		
C Max FLS$^{\neq;\geq}$	Apx-complete (within 2)	
C Max FLS$^{\mathcal{R};\neq}$	trivial	

Table 3.2: Main approximability results for Max FLS variants with $\mathcal{R} \in \{=, \geq, >, \neq\}$. The constrained versions C Max FLS$^{=;\mathcal{R}}$ with real variables and mandatory equations are equivalent to the corresponding Max FLS$^{\mathcal{R}}$. Every nonstrict inequality can be replaced by strict inequality.

Table 3.2 summarizes our main approximability results. All of them hold for inhomogeneous systems with integer coefficients and no pairs of identical relations, but most of them are still valid for homogeneous systems with ternary coefficients. In order to avoid trivial solutions we required the variables occurring most frequently in the satisfied relations to be nonzero.

As seen in the table, the approximability of similar variants of MAX FLS can differ enormously depending on the type of relational operators. Nevertheless, there is some structure: all basic versions of MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ are APX-hard, restricting the variables to binary (bipolar) values or introducing a set of relations that must be satisfied makes them harder to approximate, and if both restrictions are considered simultaneously all problems become NPO PB-complete. The case of MAX FLS$^{\neq}$ is considerably different. Its constrained variants are intrinsically easier than the corresponding problems with the other types of relations except when constraints are imposed both on the relations and the variables.

While MAX FLS$^{=}$ is not approximable within p^{ε} for some $\varepsilon > 0$, MAX FLS$^{=}$ over $GF(q)$ can be approximated within q but not within q^{ε} for some $\varepsilon > 0$, and MAX FLS$^{=}$ restricted to nonnegative variables is MAX IND SET-hard. Moreover, all approximability results for basic MAX FLS with any type of relational operator do also hold for the weighted versions with polynomially bounded weights. It is worth noting that some problems, like BIP MAX FLS$^{>}$ and BIP MAX FLS$^{\geq}$, become harder to approximate when inhomogeneous systems are considered.

We should point out that Arora, Babai, Stern and Sweedyk simultaneously addressed the hardness of approximating one variant of this broad class of problems, namely MAX FLS$^{=}$ [18]. By delving into the interactive proof connection, they independently showed that this problem cannot be approximated within any constant factor unless P = NP, and within a factor $2^{\log^{0.5-\varepsilon} n}$ for any $\varepsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$), where n is the number of variables or equations. But the result in theorem 3.2.4 is stronger because, on one hand, for large n the factor $2^{\log^{0.5-\varepsilon} n}$ is smaller than n^{δ} for any fixed $\delta > 0$ and, on the other hand, P $\neq$ NP is more likely to be true than NP $\not\subseteq$ DTIME($n^{\text{polylog}\, n}$).

Several interesting questions are still open. Are there better approximation algorithms for MAX FLS$^{>}$ and MAX FLS$^{\geq}$? Does MAX FLS$^{=}$ become harder when the variables are constrained to be nonnegative or is it already MAX IND SET-hard? Are the problems we have shown MAX IND SET-hard really NPO PB-complete?

3.3 The hardness of approximating Min ULR

We now turn to the approximability of the complementary minimization problems Min ULR where the objective is to minimize the number of unsatisfied relations instead of maximizing the number of satisfied ones. In other words, we want to remove the smallest number of relations from an infeasible linear system to achieve feasibility. In the process of studying the approximability of Min ULR we shall derive strong bounds on the approximability of a class of closely related problems which consist, given a feasible linear system, of finding a solution with as few nonzero variables as possible.

In section 3.3.1 we mention the hardness result for Min ULR$^=$ obtained in [18] and we determine alternative bounds on the approximability of the basic versions Min ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{\geq, >\}$. In section 3.3.2 we study the approximability of the above-mentioned class of problems concerning feasible systems and discuss its relationship with Min ULR. As we shall see in chapter 4, a particular case of this problem arises when designing close-to-minimal feedforward neural networks of linear threshold units. Section 3.3.3 is devoted to the weighted and constrained variants of Min ULR. The next section examines the approximability of these problems when the variables are restricted to take a finite number of discrete values. Finally, in section 3.3.5 the various results are summarized and a few open questions are mentioned.

3.3.1 Basic Min ULR

In this section we focus on the basic versions of Min ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$. As previously seen, Min ULR$^{\neq}$ is trivially solvable because any such system is feasible.

In homogeneous Min ULR$^=$ and Min ULR$^{\geq}$, we are only interested in solutions where the variable(s) occurring in the largest number of satisfied relations are nonzero. This rules out the trivial $\mathbf{x} = \mathbf{0}$ solution as well as those obtained by setting all variables to zero except one of the variables occurring in the smallest number of equations.

Before establishing strong bounds on the approximability of Min ULR$^{\geq}$ and Min ULR$^>$, we point out the relationship between the approximability of Min ULR$^{\geq}$ and that of Min ULR$^=$.

Proposition 3.3.1 Min ULR$^{\geq}$ *is at least as hard to approximate as* Min ULR$^=$ *but not harder than* Min ULR$^=$ *with nonnegative variables.*

Proof We first reduce Min ULR$^=$ to Min ULR$^{\geq}$ and then reduce Min ULR$^{\geq}$ to Min ULR$^=$ with nonnegative variables.

Let $(A, \mathbf{b})$ be an arbitrary instance of MIN ULR$^=$. For each equation $\mathbf{a}^i\mathbf{x} = b_i$ where $\mathbf{a}^i$ denotes the ith row of A, $1 \leq i \leq p$, we consider the two inequalities $\mathbf{a}^i\mathbf{x} \geq b_i$ and $-\mathbf{a}^i\mathbf{x} \geq -b_i$. Since any solution $\mathbf{x}$ satisfies at least one inequality of each such pair, finding a solution that minimizes the number of unsatisfied equations in $A\mathbf{x} = \mathbf{b}$ is equivalent to finding a solution that minimizes the number of unsatisfied inequalities of the corresponding system of size $2p \times n$.

Let $(A, \mathbf{b})$ be an arbitrary instance of MIN ULR$^\geq$. By replacing each variable x_i unrestricted in sign by the difference $x'_i - x''_i$ of two nonnegative variables $x'_i, x''_i \geq 0$ and by adding a slack variable for each inequality, we obtain an equivalent system with p equations and $2n + p$ nonnegative variables. □

Proposition 3.3.2 MIN ULR$^\geq$ *and* MIN ULR$^>$ *with integer coefficients are equivalent.*

PROOF We simply use the fact that any system $A\mathbf{x} \leq \mathbf{b}$ has a solution if and only if the system $A\mathbf{x} < \mathbf{b} + \varepsilon\mathbf{1}$ has a solution, where $\varepsilon = 2^{-2L}$ and L is the size (in bits) of the binary encoded input instance [137]. □

In [18] Arora et al. established the same type of approximability bound for MIN ULR$^=$ as for MAX FLS$^=$. More precisely, they showed that the former problem cannot be approximated within any constant unless P = NP and within a factor of $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$), where n is the size of the input instance. According to propositions 3.3.1 and 3.3.2, this is also valid for MIN ULR with strict and nonstrict inequalities.

We now prove a nonapproximability result for MIN ULR$^\geq$ and MIN ULR$^>$ which is more likely to be true but whose bound is not as strong.

Theorem 3.3.3 MIN ULR$^\geq$ *and* MIN ULR$^>$ *are* MIN DOMINATING SET*-hard. They cannot be approximated within any constant unless* P = NP, *and within* $c \log n$ *for any* $c < 1/8$ *unless* NP $\subseteq$ DTIME($n^{\log\log n}$), *even when restricted to homogeneous systems with ternary coefficients in* $\{-1, 0, 1\}$.

PROOF We proceed by cost preserving reduction from MIN DOMINATING SET which is defined as follows. Given an undirected graph $G = (V, E)$, find a minimum cardinality set $V' \subseteq V$ that dominates all nodes of G, i.e. for all $v \in V \setminus V'$ there exists $v' \in V'$ such that $[v, v'] \in E$.

Let $G = (V, E)$ be an arbitrary instance of MIN DOMINATING SET. For each node $v_i \in V$, $1 \leq i \leq n$, we consider the homogeneous inequality

$$x_i \geq 0 \tag{3.53}$$

and the inhomogeneous inequality

$$x_i + \sum_{j \in N(v_i)} x_j \leq -1. \tag{3.54}$$

where j belongs to $N(v_i)$ if and only if v_j is adjacent to v_i. Thus we have a system with $2n$ inequalities and n variables.

We claim that there exists a dominating set in G of size at most s if and only if there exists a solution $\mathbf{x}$ that violates at most s inequalities of the corresponding system.

Given a dominating set $V' \subseteq V$ of size s, the solution $\mathbf{x}$ defined by

$$x_i = \begin{cases} -1 & \text{if } v_i \in V' \\ 0 & \text{otherwise} \end{cases}$$

satisfies all inhomogeneous inequalities and $n - s$ homogeneous ones.

Conversely, given a solution vector $\mathbf{x}$ that violates s inequalities, we can always satisfy every inhomogeneous inequality that is not already satisfied by making one variable x_i in the inequality negative enough. This operation yields a solution which satisfies at least as many inequalities as $\mathbf{x}$. Consider the set of nodes $V' \subseteq V$ containing all nodes v_i such that $x_i < 0$. V' is clearly a dominating set of size s, because $x_i + \sum_{j \in N(v_i)} x_j \leq -1$ only when at least one of the variables is negative, which corresponds to the case where at least one of the nodes is in the dominating set.

Since the above reduction is cost preserving and without amplification, we have exactly the same nonapproximability bounds for MIN ULR$^{\geq}$ and MIN ULR$^{>}$ as for MIN DOMINATING SET.

This reduction can easily be adapted to MIN ULR$^{>}$ restricted to homogeneous systems. It suffices to replace $\geq$ by $>$ and the -1 in the right hand side of the second type of inequalities by 0.

There is a standard way to transform an inhomogeneous MIN ULR$^{=}$ or MIN ULR$^{\geq}$ system to a homogeneous one. Given a system consisting of p relations, we first multiply all constant right hand sides by a new variable x_0. Then we only have to ensure that this variable is nonzero (for the equality case) or positive (for the inequality case). Therefore we introduce new relations $k \cdot x_0 = k \cdot y_0$ or $k \cdot x_0 \geq 0$, for $k \in [1..p+1]$, where y_0 is a new variable. In every solution that satisfies more than p relations, at least one of the new relations must be satisfied, and since all the new relations are satisfied simultaneously, x_0 will become the variable occurring in the largest number of satisfied equations. Hence x_0 is nonzero. For the inequality case this implies that $x_0 \geq 0$.

Since the same trick can also be applied to all variants of Min ULR$^=$ or Min ULR$^\geq$, we will not give this reduction every time it is used. □

Clearly, for large n and small $\varepsilon > 0$, a factor of $2^{\log^{1-\varepsilon} n}$ is larger than $c \log n$ with any $c < 1/8$, but NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$) is more likely to be true than NP $\subseteq$ DTIME($n^{\log\log n}$). Furthermore, the above proof is much simpler than that in [18].

Unlike for Max FLS$^=$, for Min ULR$^=$ we can guarantee a performance ratio that is linear in the number of variables.

Proposition 3.3.4 Min ULR$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is approximable within* $n+1$*, where* n *is the number of variables.*

Proof When applied to linear systems, Helly's theorem (see [44]) implies that, for any infeasible system of inequalities or equations in n variables, all minimal infeasible subsystems contain at most $n+1$ relations. Such an IIS, which is also called *Helly obstruction*, can be found using any polynomial time LP method [55]. According to Farkas' lemma [151], a system $A\mathbf{x} \leq \mathbf{b}$ with p inequalities and n variables is infeasible if and only if there exists a nonnegative vector $\mathbf{y} \geq \mathbf{0}$ such that $\mathbf{y}^t A = \mathbf{0}$ and $\mathbf{y}^t \mathbf{b} < 0$. In fact, the result is still valid if the vector $\mathbf{y} \geq \mathbf{0}$ is required to have at most $n+1$ nonzero components. For infeasible systems, a polynomial time LP algorithm produces a $\mathbf{y}$ satisfying Farkas' lemma. If $\mathbf{y}$ has more than $n+1$ nonzero components, some of them can be driven to zero. Therefore it suffices to find a nontrivial solution $\mathbf{z}$ of the auxiliary system $\mathbf{z}^t[A|\mathbf{b}] = [\mathbf{0}^t|0]$ such that $\mathbf{z}$ is zero for every components where $\mathbf{y}$ is zero. This simply amounts to determine a nontrivial solution to $n+1$ homogeneous equations in more than $n+1$ variables. Subtracting a multiple of $\mathbf{z}$ from $\mathbf{y}$ leads to a new $\mathbf{y}$ with fewer nonzero components. By repeating this process, we obtain in polynomial time a $\mathbf{y}$ with at most $n+1$ nonzero components that correspond to the inequalities in a Helly obstruction.

Thus starting with an infeasible system we can identify an obstruction and delete it iteratively until the resulting system is feasible, that is at most $p/(n+1)$ times. Clearly, we remove at most $n+1$ times more inequalities than needed because at each step we delete at most $n+1$ relations corresponding to a Helly obstruction while a single one may suffice. □

The question of whether a logarithmic performance ratio $c \log n$ for some $c \geq 1/8$ can be guaranteed in polynomial time unless P = NP is still open.

3.3.2 Min RVLS: a class of closely related problems

In this section we consider a class of problems pertaining to feasible systems but which are closely related to Min ULR. Given any feasible system of linear relations, we are interested in minimizing the number of Relevant Variables in the Linear System.

> Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$: Given a feasible system of linear relations $A\mathbf{x} \,\mathcal{R}\, \mathbf{b}$ with rational coefficients, find a solution satisfying all relations with as few nonzero variables as possible.

Min RVLS$^=$ is referred to as *minimum weight solution to linear equations* in Garey and Johnson's reference book [66].

There is a simple relation between the approximability of Min ULR and that of Min RVLS.

Proposition 3.3.5 Min ULR$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is at least as hard to approximate as* Min RVLS$^{\mathcal{R}}$ *with the same type of relations.*

Proof It suffices to note that for any instance of Min RVLS$^=$ given by $A\mathbf{x} = \mathbf{b}$ one can construct an equivalent instance of Min ULR$^=$. This is simply achieved by considering, for each variable x_i with $1 \leq i \leq n$, the equation

$$x_i = 0 \tag{3.55}$$

and by eliminating variables in this system using the set of equations $A\mathbf{x} = \mathbf{b}$ that must be satisfied.

Similarly, by replacing each equation (3.55) by two nonstrict inequalities one shows that Min ULR$^{\geq}$ is at least as hard to approximate as Min RVLS$^{\geq}$.

Since by proposition 3.3.2 Min ULR$^{>}$ is equivalent to Min ULR$^{\geq}$ for systems with integer coefficients, Min ULR$^{>}$ is also at least as hard as Min RVLS$^{\geq}$. □

Thus any nonapproximability bound for Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ implies an equivalent bound for the corresponding Min ULR$^{\mathcal{R}}$.

Proposition 3.3.6 MIN RVLS$^=$ *cannot be approximated within any constant factor unless* P = NP.

PROOF By reduction from a variant of SET COVERING with disjoint sets. In SET COVERING, given a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of subsets of a finite set S, one seeks a subcollection $\mathcal{C}' = \{C_{j1}, \ldots, C_{j_m}\} \subseteq \mathcal{C}$ of minimum cardinality such that $\cup_{i=1}^m C_{j_i} = S$ with $m \leq n$. Any such $\mathcal{C}'$ is a cover of S. If all the sets in $\mathcal{C}'$ are pairwise disjoint, it is an exact cover.

The proof is based on the following result by Bellare et al. on exact covers [32]. For every $c > 1$, there exists a polynomial time reduction that transforms any instance ϕ of the satisfiability problem SAT (see the Appendix) into an instance of SET COVERING with a positive integer K such that

- if ϕ is satisfiable there exists an exact cover $\mathcal{C}'$ of size K,
- if ϕ is unsatisfiable no set cover has size less than $\lfloor c \cdot K \rfloor$.

For any such instance $(S, \mathcal{C})$ of SET COVERING, we can construct a system with n variables and $|S|$ equations

$$A\mathbf{x} = \mathbf{1}, \tag{3.56}$$

where $a_{ij} = 1$ if the ith element of S belongs to C_j and 0 otherwise. $\mathbf{1}$ denotes the $|S|$-dimensional vector with all 1 components. Clearly, the nonzero variables in any solution $\mathbf{x}$ of the above system define a set cover. Conversely, given any exact cover $\mathcal{C}'$ of cardinality K, the vector $\mathbf{x}$ given by

$$x_j = \begin{cases} 1 & \text{if } C_j \in \mathcal{C}' \\ 0 & \text{otherwise} \end{cases}$$

satisfies all equations and has K nonzero variables. Thus the minimum number of nonzero variables in a solution of $A\mathbf{x} = \mathbf{1}$ is either K or at least $\lfloor c \cdot K \rfloor$.

Note that while there is a one-to-one correspondence between the exact covers and the solutions of (3.56) with 0 or 1 components, minimum cardinality covers do not necessarily correspond to a solution. For example, in the set covering instance associated to

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

all subcollections of cardinality 2 are minimum (nonexact) covers but the system has the unique solution $(1/2, 1/2, 1/2)$ with three nonzero variables. For infeasible systems like

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

the minimum number of nonzero variables can be considered as larger than n. □

According to proposition 3.3.5, the above result directly implies that MIN ULR$^=$ cannot be approximated within any constant factor unless P = NP, which has been established in [18]. Furthermore, propositions 3.3.1 and 3.3.2 show that MIN ULR$^\geq$ and MIN ULR$^>$ are at least as hard to approximate.

In fact, MIN RVLS$^=$ and MIN RVLS with inequalities are not approximable within a factor of $2^{\log^{0.5-\varepsilon} n}$ for any $\varepsilon > 0$. In order to prove this strong nonapproximability bound we consider a natural variant of the LABEL COVER problem studied in the above reference.

Definition 3.3.1 Let $G = (V_1, V_2, E)$ be a bipartite graph, B_1 and B_2 two sets of possible labels for the nodes in V_1 and V_2 respectively, and $\Pi \subseteq E \times B_1 \times B_2$ a relation consisting of admissible pairs of labels for each edge. A *labelling* of G is a function $\mathcal{P} : V_1 \cup V_2 \to 2^{B_1} \cup 2^{B_2}$ that assigns a set of possible labels to each node of G. We say that a labelling $\mathcal{P}$ *covers* an edge $e = [v_1, v_2]$ if the two sets of labels $\mathcal{P}(v_1)$ and $\mathcal{P}(v_2)$ are nonempty and for every $b_2 \in \mathcal{P}(v_2)$ there exists an admissible label $b_1 \in \mathcal{P}(v_1)$ such that $(e, b_1, b_2) \in \Pi$. A labelling that covers all edges of G is called a *total cover*. Furthermore the *cost* of a cover $\mathcal{P}$ is defined as $\sum_{v \in V_1 \cup V_2} |\mathcal{P}(v)|$. The TOTAL LABEL COVER problem is then to find a minimum cost cover. The *size* of an instance of TOTAL LABEL COVER is the total length of a description of the sets G, B_1, B_2 and Π.

The only difference between TOTAL LABEL COVER and LABEL COVER is in the objective function. In TOTAL LABEL COVER the cost is equal to the total number of labels needed to label all the nodes of the bipartite graph (instead of only considering those in V_1).

Lemma 3.3.7 *For any fixed $\varepsilon > 0$, there exists a quasi-polynomial reduction*[1] *that reduces an instance ϕ of* SAT *of size s to an instance of* TOTAL LABEL COVER *of size s', with $s' \leq 2^{\text{polylog } s}$, such that*

[1]A reduction is said quasi-polynomial if it has an $O(s^{\text{polylog } s})$ time-complexity.

- *if ϕ is satisfiable there is a total cover with cost $|V_1| + |V_2|$,*
- *if ϕ is unsatisfiable, then any labelling covering more than half of the edges has cost greater or equal to $2^{\log^{0.5-\varepsilon} s'} \cdot (|V_1| + |V_2|)$.*

Proof By construction, the bipartite graphs derived from Feige and Lovász's 2-prover 1-round interactive proof systems have the following properties (see [59, 120]). The graphs are regular (every node has the same degree), $|V_1| = |V_2|$ and for any e and $b_1 \in B_1$ there is at most one $b_2 \in B_2$ such that $(e, b_1, b_2) \in \Pi$. Furthermore, we may assume that the relation Π is restricted to have only valid labels for any node in V_1, where b_1 is *valid* for $v_1 \in V_1$ if, for every edge e incident to v_1, there is a label b_2 such that $(e, b_1, b_2) \in \Pi$. In fact, the total label covers corresponding to any satisfiable ϕ use only one label per node of the graph. Thus one easily verify that the gap for Label Cover implies an equivalent gap for Total Label Cover. □

Theorem 3.3.8 Min RVLS$^{\mathcal{R}}$ *with $\mathcal{R} \in \{=, \geq, >\}$ cannot be approximated within a factor of $2^{\log^{0.5-\varepsilon} n}$ for any $\varepsilon > 0$ unless* NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$, *where n is the number of variables.*

Proof For any Total Label Cover instance (G, B_1, B_2, Π) arising from a Feige-Lovász interactive proof system, one can construct a particular instance of Min RVLS$^=$ with exactly the same objective function. Therefore we can use the first part of the construction described in [18].

The idea is to introduce one variable for each pair $[v_i, b_i]$ where $v_i \in V_i$ and $b_i \in B_i$, $i = 1, 2$, which satisfy the following conditions. While every pair (v_2, b_2) with $v_2 \in V_2$ and $b_2 \in B_2$ is considered, only the pairs (v_1, b_1) where $v_1 \in V_1$, $b_1 \in B_1$ and v_1 is valid are retained. For each pair e, $b_1 \in B_1$ the unique label $b_2 \in B_2$ such that $(e, b_1, b_2) \in \Pi$ is denoted by $b_2[e, b_1]$. We construct an inhomogeneous system

$$A\mathbf{x} = \mathbf{1} \tag{3.57}$$

with $|E|(|B_2| + 1)$ equations, namely $|B_2| + 1$ for each edge $e \in E$. For each column of A, the components corresponding to a given e are called its *e-projection*. The columns are defined as follows. For each pair $v_2 \in V_2$, $b_2 \in B_2$ the e-projection is $\mathbf{u}_j$ (the $|B_2| + 1$-dimensional vector in which the jth component is 1 and all the others 0) and $\mathbf{0}$ otherwise, where $\mathbf{0}$ denotes the trivial vector with all zero components. For each valid pair $v_1 \in V_1$,

$b_1 \in B_1$ the e-projection is $\mathbf{1} - \mathbf{u}_{b_2[e,b_1]}$ if e is incident to v_1 and $\mathbf{0}$ otherwise. Thus we have a linear system with $|E|(|B_2|+1)$ equations and at most $|V_1| \cdot |B_1| + |V_2| \cdot |B_2|$ variables.

It is easily verified that the nonzero variables of any solution $\mathbf{x}$ of (3.57) define a total cover of the original bipartite graph G. Thus, according to lemma 3.3.7, MIN RVLS$^=$ cannot be approximated within the given factor unless NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$.

By inspecting the construction, one observes that the same bound holds even for homogeneous systems with binary coefficients. Since any equation can be replaced by two nonstrict inequalities, the result is also valid for MIN RVLS$^\geq$. Using the equivalence between systems with nonstrict and strict inequalities, we obtain the same bound for MIN RVLS$^>$. □

This nonapproximability result can be still strengthened.

Theorem 3.3.9 MIN RVLS$^\mathcal{R}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is not approximable in polynomial time within a factor of* $2^{\log^{1-\varepsilon} n}$ *for any* $\varepsilon > 0$ *unless* NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$, *where* n *is the number of variables.*

PROOF We proceed by self-improvement as in [19]. The idea is to consider the reduction in proposition 3.3.6 from SET COVERING to MIN RVLS with a fixed constant gap $c > 1$ between the satisfiable and unsatisfiable cases and to increase it recursively. For any particular SET COVERING instance, we start with the corresponding system (3.56) whose coefficients are 0 or 1 and we construct the squared system

$$A'\mathbf{x}' = \mathbf{1}' \tag{3.58}$$

obtained by replacing each 1 coefficient of A by the whole matrix A and each 0 coefficient by the $p \times n$ matrix with all 0. Thus A' is a matrix of size $p^2 \times n^2$ and $\mathbf{1}'$ is a vector with p^2 components all equal to 1. Clearly, if the corresponding instance ϕ of SAT is satisfiable there is a solution $\mathbf{x}'$ of (3.58) with K^2 variables equal to 1 and all the other to 0. On the contrary, if ϕ is unsatisfiable any solution $\mathbf{x}'$ that satisfies all equations of (3.58) has at least $\lfloor c^2 K^2 \rfloor$ nonzero variables.

By applying this construction t times recursively, we obtain a system with p^{2^t} equations and n^{2^t} variables. Let us take $t = \lfloor \log(\log^\beta n) \rfloor$, where n is the number of variables in the MIN RVLS$^=$ instance corresponding to the considered SAT instance ϕ, and β is a positive real number. The construction requires $O(n^{\text{polylog}\, n})$ time because the system has $p' = p^{O(\log^\beta n)}$ equations and $n' = n^{O(\log^\beta n)} = 2^{O(\log^{\beta+1} n)}$ variables. Since $\log n' = O(\log^{\beta+1} n)$, we obtain a gap of $c^{2^t} = c^{\log^\beta n} = 2^{O(\log n')^{\beta/(\beta+1)}}$.

The theorem follows by contradiction. Suppose there exists a polynomial time algorithm that approximates Min RVLS$^=$ instances with n variables within a factor $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 1/(\beta+1)$. By applying it to the resulting instance of Min RVLS$^=$, one could decide in $O(n^{\text{polylog}\, n})$ time whether any given instance ϕ of Sat is satisfiable. But this would imply that NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$.

The strict and nonstrict inequality cases are dealt with similarly. □

Since in the reduction the number of examples p is polynomially related to the number of components n, the same bound is also valid when n is replaced by p.

According to proposition 3.3.5, this result for Min RVLS$^=$ immediately implies the nonapproximability of Min ULR$^\mathcal{R}$ with $\mathcal{R} \in \{=, \geq, >\}$ within the same factor.

In fact, there is a reduction from homogeneous Min ULR$^\mathcal{R}$ to Min RVLS$^\mathcal{R}$ with $\mathcal{R} \in \{=, \geq\}$.

Proposition 3.3.10 Min ULR$^\mathcal{R}$ *with* $\mathcal{R} \in \{=, \geq\}$ *is as hard to approximate as* Min RVLS$^\mathcal{R}$ *with the same type of relations.*

Proof Min RVLS$^\mathcal{R}$ with $\mathcal{R} \in \{=, \geq\}$ can be reduced to Min ULR$^\mathcal{R}$ by proposition 3.3.5, and Min ULR$^\mathcal{R}$ can be transformed to homogeneous Min ULR$^\mathcal{R}$ as shown in the proof of theorem 3.3.3. Thus it suffices to show that homogeneous Min ULR$^\mathcal{R}$ can be reduced to Min RVLS$^\mathcal{R}$ for $\mathcal{R} \in \{=, \geq\}$.

Let $\sum_{j=1}^n a_{ij} x_j = 0$ with $1 \leq i \leq p$ be an instance of homogeneous Min ULR$^=$. We construct an instance of Min RVLS$^=$ consisting of $p(n+1)$ equations of the following type:

$$\sum_{j=1}^n a_{ij} x_j = y_{ik}$$

where $1 \leq i \leq p$, $1 \leq k \leq n+1$ and y_{ik} are $p(n+1)$ new variables.

If s equations of $A\mathbf{x} = \mathbf{0}$ are satisfied, then we will get a solution of Min RVLS$^=$ with between $s(n+1)$ and $s(n+1)+n$ variables equal to zero. Conversely, a solution of Min RVLS$^=$ with between $s(n+1)$ and $s(n+1)+n$ variables equal to zero, will give us a solution of Min ULR$^=$ with s satisfied equations. The reduction is an S-reduction with size amplification $O(pn)$.

The same construction can be used for Min RVLS$^\geq$, just by substituting equations with nonstrict inequalities. □

Note that due to the size amplification the nonapproximability bound for Min ULR$^=$ established in [18] leads to a smaller bound for Min RVLS$^=$ than that of theorem 3.3.9.

Using the same reduction, the complementary maximization problem Max IVLS$^=$, Maximum number of Irrelevant Variables in Linear Systems, can be shown in its homogeneous version to be equally hard to approximate as homogeneous Max FLS$^=$, i.e. not approximable within p^ε for some $\varepsilon > 0$ unless P = NP [10].

Interestingly, Max IVLS$^\geq$ and Max IVLS$^>$ are much harder to approximate than Max FLS$^\geq$ and Max FLS$^>$, respectively. While the latter problems are Apx-complete and can be approximated within 2, it is easily proved that the former ones are Max Ind Set-hard. It suffices to construct, for each edge $e = [v_i, v_j]$, the inequality $x_i + x_j \geq 1$ or $x_i + x_j > 0$ and to observe that there is a correspondence between the independent sets of cardinality at least s and the solutions with at least s zero components.

3.3.3 Weighted and constrained Min ULR

This section is devoted to the weighted variants of Min ULR where a weight is assigned to each relation and the goal is to find a solution that minimizes the total weight of the unsatisfied relations.

As for Max FLS, the constrained versions with some mandatory relations are important special cases of weighted Min ULR. C Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ with $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$ denotes the variant where the mandatory relations are of type $\mathcal{R}_1$ and the optional ones of type $\mathcal{R}_2$. When $\mathcal{R}_1 = \mathcal{R}_2$ the problem can be seen as a weighted Min ULR$^{\mathcal{R}_1}$ problem in which the weight of every mandatory relation is larger than the total weight of all optional ones.

Proposition 3.3.11 *Weighted* Min ULR$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *and positive integer weights bounded by a polynomial in the input size are as hard to approximate as the basic versions.*

Proof Same argument as in the proof of proposition 3.2.9 concerning Max FLS. □

This equivalence between weighted and unweighted variants holds also for the constrained versions with the same type of optional and mandatory relations. To make sure that the mandatory relations are satisfied, a large enough number of equivalent relations is required, namely more than the number of optional ones. The approximability of constrained MIN ULR is thus very well characterized by the approximability of the unconstrained versions.

It is worth noting that no such relation exists between constrained and unweighted versions of the complementary problems MAX FLS. As shown in section 3.2.3, enforcing some mandatory relations makes MAX FLS with inequalities harder to approximate. While MAX FLS$^>$ and MAX FLS$^\geq$ are APX-complete, the constrained variants are MAX IND SET-hard.

Any instance of a constrained problem C MIN ULR$^{=;\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$ can be transformed into an equivalent instance of MIN ULR$^{\mathcal{R}}$ by eliminating variables in the set of optional relations using the set of mandatory ones. All variants C MIN ULR$^{\mathcal{R};\neq}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$ are trivial for the same reason as MIN ULR$^{\neq}$.

Proposition 3.3.12 C MIN ULR$^{\geq;=}$ *is as hard to approximate as* MIN ULR$^{\geq}$*.*

PROOF According to proposition 3.3.1, MIN ULR$^{\geq}$ can be reduced to MIN ULR$^{=}$ with nonnegative variables. This can be expressed as a C MIN ULR$^{\geq;=}$ problem where the mandatory inequalities ensure that all variables are nonnegative.

Conversely, C MIN ULR$^{\geq;=}$ can be reduced to C MIN ULR$^{\geq;\geq}$ by substituting each optional equation by two inequalities as in the proof of proposition 3.3.1. We can then use the standard reduction from C MIN ULR$^{\geq;\geq}$ to MIN ULR$^{\geq}$. □

Similarly we can show that C MIN ULR$^{>;=}$ is at least as hard to approximate as MIN ULR$^{>}$.

Proposition 3.3.13 C MIN ULR$^{\neq;=}$ *is* MIN DOMINATING SET*-hard. Therefore it cannot be approximated within any constant unless* P = NP *and within* $c \log n$ *for any* $c < 1/8$ *unless* NP $\subseteq$ DTIME$(n^{\log\log n})$*, even when restricted to homogeneous systems with binary coefficients.*

PROOF The proof is by reduction from MIN DOMINATING SET. Let $G = (V, E)$ be an arbitrary instance of MIN DOMINATING SET. For each node $v_i \in V$, $1 \leq i \leq n$, we consider the optional equation

$$x_i = 0 \tag{3.59}$$

and the mandatory inhomogeneous relation

$$x_i + \sum_{j \in N(v_i)} x_j \neq 0 \tag{3.60}$$

where j belongs to $N(v_i)$ if and only if v_j is adjacent to v_i.

We claim that there exists a dominating set in G of size at most s if and only if there exists a solution that violates at most s relations of the corresponding linear system.

Given a dominating set $V' \subseteq V$ of size s, the solution $\mathbf{x}$ defined by

$$x_i = \begin{cases} 1 & \text{if } v_i \in V' \\ 0 & \text{otherwise} \end{cases}$$

satisfies all mandatory relations and all but s of the optional equations.

Conversely, given a solution vector $\mathbf{x}$ that satisfies all mandatory relations and violates s of the optional relations, we consider the set of nodes $V' \subseteq V$ containing all nodes v_i such that $x_i \neq 0$. V' is clearly a dominating set of size s, because $x_i + \sum_{j \in N(v_i)} x_j \neq 0$ only when at least one of the variables is nonzero, which corresponds to the case where at least one of the nodes is in the dominating set. $\square$

3.3.4 Variants with bounded discrete variables

In this section we determine the approximability of Min ULR$^{\mathcal{R}}$ and Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$ when the variables are restricted to take binary values in $\{0, 1\}$ or bipolar ones in $\{-1, 1\}$. The corresponding variants of Min ULR and Min RVLS are referred to as Bin Min ULR, Bip Min ULR, Bin Min RVLS and Bip Min RVLS respectively.

Theorem 3.3.14 *For every combination of* $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$*,* Bin Min ULR$^{\mathcal{R}_1}$ *and* C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ *are* NPO PB*-complete. Assuming* P$\neq$NP*,* C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ *cannot be approximated within* $s^{1-\varepsilon}$ *for any* $\varepsilon > 0$ *and* Bin Min ULR$^{\mathcal{R}_1}$ *cannot be approximated within* $s^{0.5-\varepsilon}$ *for any* $\varepsilon > 0$*, where* $s = \max\{n, p\}$ *i.e. the maximum of the number of variables and the number of relations.*

Proof We will first show the result for the problem C Bin Min ULR$^{\geq;\geq}$ and then extend the result to the other variants.

We proceed by reduction from Min Ind Dom Set where, given an undirected graph $G = (V, E)$, one looks for a minimum cardinality independent set $V' \subseteq V$ that dominates all nodes of G.

Let $G = (V, E)$ be an arbitrary instance of Min Ind Dom Set. For each node $v_i \in V$, $1 \leq i \leq n$, we consider the optional inequality

$$x_i \leq 0 \tag{3.61}$$

and the inhomogeneous mandatory inequality

$$x_i + \sum_{j \in N(v_i)} x_j \geq 1 \tag{3.62}$$

where j belongs to $N(v_i)$ if and only if v_j is adjacent to v_i. Furthermore, we construct for each edge $[v_i, v_j] \in E$ the inhomogeneous mandatory inequality

$$x_i + x_j \leq 1. \tag{3.63}$$

Thus we have a system with n variables, n optional inequalities and $n + |E|$ mandatory ones.

We claim that there exists an independent dominating set in G of size at most s if and only if there exists a solution $\mathbf{x} \in \{0,1\}^n$ that violates s relations of the corresponding linear system.

Given an independent dominating set $V' \subseteq V$ of size s, the solution $\mathbf{x}$ defined by

$$x_i = \begin{cases} 1 & \text{if } v_i \in V' \\ 0 & \text{otherwise} \end{cases}$$

satisfies all mandatory inequalities and $n - s$ of the optional ones.

Conversely, given a solution vector $\mathbf{x}$ that violates s of the relations, we consider the set of nodes $V' \subseteq V$ containing all nodes v_i such that $x_i = 1$. V' is clearly of size s, independent (because of the mandatory relations (3.63)) and dominating (because of the mandatory relations (3.62)).

The theorem follows because Min Ind Dom Set is NPO PB-complete and cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$, where n is the sum of the number of nodes and edges in the graph.

For the other constrained problems C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$, we use the same reduction but the right hand side of the three types of relations must be substituted according to table 3.3. In the case of mandatory equations we need to introduce $2|E| - n$ additional slack variables y_{ij} and $|E|$ additional slack variables z_{ij}. Thus the total number of variables will be $3|E|$, that is, still a linear number in n and $|E|$.

To get the hardness results for the unconstrained problems, we add $|V| + 1$ copies of each mandatory relation so that they are more valuable than the optional ones. Since such a reduction has a quadratic size amplification, we get a weaker nonapproximability bound than for Min Ind Dom Set. □

	operator $\geq$	operator $>$	operator $\neq$	operator $=$
type (3.61)	≤ 0	< 1	$\neq 1$	$= 0$
type (3.62)	≥ 1	> 0	$\neq 0$	$= \lvert N(v_i)\rvert - \sum_{j=1}^{\lvert N(v_i)\rvert-1} y_{ij}$
type (3.63)	≤ 1	< 2	$\neq 2$	$= 1 - z_{ij}$

Table 3.3: Right hand side values used in the C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ reductions for all the possible choices of $\mathcal{R}_1$ and $\mathcal{R}_2$. The three types of optional and mandatory relations are mentioned.

It is worth noting that Bin Min ULR$^{\mathcal{R}_1}$ and C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ with $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$ remain NPO PB-complete for homogeneous systems. In the above reduction from Min Ind Dom Set, we multiply each nonzero constant in the right hand side of a relation by a new variable x_0 which should be viewed as the constant 1. In order to prevent x_0 from being zero we introduce new mandatory relations $x_0 \geq 0$, $x_0 > 0$, $x_0 \neq 0$, or $x_0 = y_0$ with a new variable y_0, depending on the type of relations. In the case of nonstrict inequalities and equalities we introduce (as in the proof of theorem 3.3.3) a lot of copies of the relations in order to make x_0 the variable occurring the most frequently in the satisfied relations.

Corollary 3.3.15 *For every combination of* $\mathcal{R}_1, \mathcal{R}_2 \in \{=, \geq, >, \neq\}$, Bip Min ULR$^{\mathcal{R}_1}$ *and* C Bip Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ *are* NPO PB*-complete. Assuming* P$\neq$NP, C Bip Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$ *cannot be approximated within* $s^{1-\varepsilon}$ *for any* $\varepsilon > 0$ *and* Bip Min ULR$^{\mathcal{R}_1}$ *cannot be approximated within* $s^{0.5-\varepsilon}$ *for any* $\varepsilon > 0$*, where* $s = \max\{n, p\}$*.*

Proof By simple cost preserving transformation from Bin Min ULR$^{\mathcal{R}_1}$ and C Bin Min ULR$^{\mathcal{R}_1;\mathcal{R}_2}$. Using the same variable substitution given in corollary 3.2.14. □

Min RVLS restricted to binary or bipolar variables turns out to be also very hard to approximate.

Proposition 3.3.16 Bin Min RVLS$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is* NPO PB*-complete. Assuming* P $\neq$ NP*, neither* Bin Min RVLS$^{\geq}$ *nor* Bin Min RVLS$^{>}$ *can be approximated within* $n^{1-\varepsilon}$ *for any* $\varepsilon > 0$*, and* Bin Min RVLS$^{=}$ *cannot be approximated within* $n^{0.5-\varepsilon}$ *for any* $\varepsilon > 0$*, where* n *is the number of variables.*

PROOF As before, we proceed by reduction from MIN IND DOM SET. The reduction is very similar to the one used in theorem 3.3.14 for proving the NPO PB-hardness of C BIN MIN ULR$^{\mathcal{R};\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$. The BIN MIN RVLS$^{\mathcal{R}}$ instance is simply composed of the mandatory relations constructed in the C BIN MIN ULR$^{\mathcal{R};\mathcal{R}}$ instance. The number of violated optional relations in the proof exactly corresponds to the number of nonzero variables. This implies that BIN MIN RVLS$^{\geq}$ and BIN MIN RVLS$^{>}$ are NPO PB-hard.

For BIN MIN RVLS$^{=}$, we still have to deal with the slack variables y_{ij} and z_{ij} that have been added. Suppose there is a total number of N slack variables.

In order to make the value of the x variables more important than the values of the N slack variables altogether, we introduce, for each variable x_i, N new variables $x_{i1}, \ldots, x_{iN}$ and the N additional equations $x_i - x_{ij} = 0$ for $j \in [1..N]$. In any solution $\mathbf{x}$ of the resulting instance we will have, for each variable x_i, that $x_i = x_{i1} = \ldots = x_{iN}$. Consider the set of nodes $V' \subseteq V$ containing all nodes v_i such that $x_i = 1$. V' is clearly independent and dominating. If t variables in $\mathbf{x}$ are equal to 1, the size of the independent set will be $\lfloor t/(N+1) \rfloor$.

Conversely, an independent dominating set containing s nodes corresponds to a solution of the BIN MIN RVLS$^{=}$ instance with between $s(N+1)$ and $s(N+1)+N$ variables equal to 1. Thus the reduction is an S-reduction with size amplification $O(nN)$ and we get the nonapproximability bound $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$, where n is the number of variables. □

Using the cost preserving transformation in the proof of corollary 3.3.15 one can show the same nonapproximability results for BIP MIN RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$.

Note that the BIN MIN RVLS$^{\geq}$ problem is equivalent to MIN PB $0-1$ PROGRAMMING, which is complete for MIN PB [100] and hence for NPO PB. Moreover, the corresponding maximization problem BIN MAX IVLS$^{\geq}$ is equivalent to MAX PB $0-1$ PROGRAMMING which is NPO PB-complete and cannot be approximated within $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$, where n is the number of variables, unless P = NP [48].

Finally, we should point out that the natural variants of basic MIN ULR, named THRESHOLD MIN ULR, where the objective function is defined with respect to a lower bound on the minimum number of unsatisfied relations (instead of zero), are at least as hard as the binary versions.

THRESHOLD MIN ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system as in MIN ULR$^{\mathcal{R}}$ and a lower bound T on the minimum number of unsatisfiable relations, find a solution $\mathbf{x} \in \mathbb{Q}^n$ that minimizes the difference between the number s of relations it violates and the bound T.

Since at least T relations are violated, the objective function $s - T$ is always nonnegative.

Theorem 3.3.17 Threshold Min ULR$^{\mathcal{R}}$ *with* $\mathcal{R} \in \{=, \geq, >\}$ *is* NPO PB-*complete. Assuming* P $\neq$ NP, Threshold Min ULR$^{\mathcal{R}}$ *cannot be approximated within* $n^{1-\varepsilon}$ *for any* $\varepsilon > 0$*, where* n *is the number of variables, and not within* $p^{0.5-\varepsilon}$ *for any* $\varepsilon > 0$*, where* p *is the number of relations.*

Proof By reduction from Bin Min ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$. Given an input instance of Bin Min ULR$^{\mathcal{R}}$ with n binary variables $x_1, \ldots, x_n$ and p relations, we construct an instance of Threshold Min ULR$^{\mathcal{R}}$ by extending the system with relations which ensure that the variables just take zero or one values.

In the equality case we add, for every variable x_i, $p + 1$ copies of the equations $x_i = 0$ and $x_i = 1$. If all variables in a solution vector take values in $\{0, 1\}$, exactly $n(p + 1)$ of the added equations will be satisfied and $n(p + 1)$ violated. For each variable that is not either zero or one, $p + 1$ additional equations will be violated, which are more than the number of equations in the original system. If we choose $n(p + 1)$ as the threshold we will get an S-reduction from Bin Min ULR$^{=}$ to Threshold Min ULR$^{=}$ without variable amplification and with a relation amplification of $O(np)$. In the proof of the nonapproximability bound of Bin Min ULR$^{\mathcal{R}}$ in theorem 3.3.14, the number of relations is about the same as the number of variables, and consequently Threshold Min ULR$^{=}$ cannot be approximated within $p^{0.5-\varepsilon}$ for any $\varepsilon > 0$, where p is the number of relations.

For Threshold Min ULR$^{\geq}$ we use the same construction, but instead of the two equations $x_i = 0$ and $x_i = 1$ we include the four inequalities $x_i \leq 0$, $x_i \geq 0$, $x_i \leq 1$, and $x_i \geq 1$. If all components of a solution vector are either zero or one, just one of these four inequalities will be violated. Otherwise two of the inequalities will be violated. The threshold and the rest of the proof is the same as for the equality case.

Finally, for Threshold Min ULR$^{>}$ we just use the equivalence between Min ULR$^{\geq}$ and Min ULR$^{>}$ mentioned in proposition 3.3.2. □

3.3.5 Summary of Min ULR and Min RVLS results

As for Max FLS, the various versions of Min ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$ that we have considered are obtained by placing constraints on the coefficients, on the variables and by assigning a possibly different importance to each relation.

Table 3.4 summarizes the nonapproximability results that hold provided P $\neq$ NP. As

	Real variables	Binary variables
Min ULR$^{=}$	not within any constant [18]	NPO PB-complete
Min ULR$^{\geq}$	Min Dominating Set-hard	
Min ULR$^{\neq}$	trivial	
C Min ULR$^{\geq;\geq}$	as hard as Min ULR$^{\geq}$	
C Min ULR$^{\geq;>}$	at least as hard as Min ULR$^{>}$	
C Min ULR$^{\geq;=}$	at least as hard as Min ULR$^{\geq}$	
C Min ULR$^{\neq;\geq}$		
C Min ULR$^{\neq;=}$	Min Dominating Set-hard	
C Min ULR$^{\mathcal{R};\neq}$	trivial	

Table 3.4: Main approximability results for Min ULR variants with $\mathcal{R} \in \{=, \geq, >, \neq\}$ that hold assuming P $\neq$ NP. The constrained versions C Min ULR$^{=;\mathcal{R}}$ with real variables and mandatory equations are equivalent to the corresponding Min ULR$^{\mathcal{R}}$. The table is still valid when nonstrict inequalities are substituted by strict inequalities and vice versa. The only difference is that C Min ULR$^{\geq;=}$ is as hard as Min ULR$^{\geq}$ while C Min ULR$^{>;=}$ is only at least as hard as Min ULR$^{>}$.

shown, the approximability of similar variants of Min ULR differs depending on the type of relational operators. The results are valid for inhomogeneous systems with integer coefficients and no pairs of identical relations, and some of them are still valid for homogeneous systems with ternary, and even binary, coefficients. In order to avoid trivial solutions for relations that are equalities or nonstrict inequalities, we required the variables occurring most frequently in the satisfied relations to be nonzero.

Min ULR$^{=}$ has recently been shown not to be approximable within any constant unless P = NP and not within $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$) [18]. Using a simple reduction from Min Dominating Set, we obtained a weaker but more likely logarithmic bound for Min ULR$^{\geq}$ as well as Min ULR$^{>}$. Moreover, we got for free the constant factors results.

In the process of studying Min ULR, we proved that the closely related class of problems Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ cannot be approximated within a constant factor and within $2^{\log^{1-\varepsilon} n}$ under the usual assumptions. Our nonapproximability result for

Min ULR$^=$, that is of interest by its own, directly implies the bound established in [18] for Min ULR$^=$. Interestingly both problems with equations or nonstrict inequalities are equivalent to approximate. However, due to the size amplification of the reduction from Min ULR$^=$ to Min RVLS$^=$ our bound for Min RVLS$^=$ is not implied by that for Min ULR$^=$.

The weighted variants of Min ULR are not harder to approximate than the unweighted ones if the weights are polynomially bounded. Restricting the variables to binary (or bipolar) values makes these problems harder to approximate, namely NPO PB-complete. As to Min ULR$^{\neq}$, while the basic version is trivial, some of the constrained variants are hard to approximate, and all binary versions turn out to be NPO PB-complete. It is worth recalling that nonapproximability bounds such as $n^{1-\varepsilon}$ for any $\varepsilon > 0$ make the existence of any nontrivial approximation algorithm extremely unlikely.

Some interesting questions concerning the approximability of Min ULR$^{\mathcal{R}}$ and Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$ are still open. Assuming P $\neq$ NP, can we guarantee a logarithmic performance ratio $c \log n$ for some $c \geq 1/8$? Do these problems admit polynomial time algorithms providing approximate solutions within $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$?

3.4 Concluding remarks

We have thoroughly investigated the complexity of the two complementary classes of problems Max FLS and Min ULR as well as of their main variants. In particular, we proved that the basic versions with $=$, $\geq$ and $>$ relations are NP-hard even for homogeneous systems with bipolar coefficients. Our set of results shows that the approximability of the Min ULR variants differs considerably from that of the complementary Max FLS problems. While all basic versions of Min ULR with equations and inequalities can be approximated within a factor $n+1$, where n is the number of variables, Max FLS$^=$ is not approximable within p^ε for some $\varepsilon > 0$, where p is the number of equations. Depending on how large p is with respect to n, this may imply a big gap between the approximability of the two complementary problems with equations. On the other hand, none of Min ULR with $\mathcal{R} \in \{=, \geq, >\}$ is approximable within a factor $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$, unless NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$, while Max FLS$^{\geq}$ and Max FLS$^{>}$ can be approximated within 2 but not within every constant.

While weighted Max FLS and Min ULR with integer and polynomially bounded weights are both as hard to approximate as the corresponding unweighted problems, an important difference occurs for the constrained variants. Imposing a set of mandatory relations makes Max FLS with inequalities much harder to approximate: they are Apx-complete and become Max Ind Set-hard. But this is not true for Min ULR since the constrained and unconstrained versions turn out to be equally hard.

When the variables are restricted to take binary or bipolar values, both classes of problems reach the highest level of complexity, namely they are NPO PB-complete.

Finally, one should recall that if Khanna's et al. claim mentioned in section 2.6 turns out to be valid, the variants we proved to be Max Ind Set-hard would be complete for the class of all problems approximable within polynomial performance ratios. In other words, they would all be NPO PB-complete.

Chapter 4

On the complexity of learning in feedforward neural networks

In this chapter we discuss the consequences of our computational complexity results for Max FLS, Min ULR and Min RVLS on the difficulty of learning in feedforward neural networks of linear threshold units. First we recall how such computational models work and what is meant by training and designing them. In section 4.2 we briefly describe the general PAC model of learning from examples that provides important insights into the training and design problems. In section 4.3 we discuss the difficulty of training single-layer networks. In particular, we disprove the "folk" theorem stating that training this type of networks is easy while it becomes intractable as soon as there are hidden units. Our discussion goes beyond artificial neural networks and leads to an extension of the PAC-learning model which makes it more realistic. Section 4.4 is devoted to the fundamental issue of designing close-to-minimum size networks. Compact networks are of particular interest because they are expected to perform well also on data which does not belong to the training set. As a consequence of our nonapproximability bound for Min RVLS, we show that no polynomial time algorithm can design a network with a number of nonzero weights that is within a factor $2^{\log^{1-\varepsilon} p}$ of the optimum for any $\varepsilon > 0$ unless $\mathrm{NP} \not\subseteq \mathrm{DTIME}(p^{\mathrm{polylog}\, p})$, where p is the number of training examples.

4.1 Feedforward networks of linear threshold units

As mentioned in section 1.2.6, we are interested in feedforward neural networks. Although such circuit-free networks may be composed of units performing continuous functions [85], we focus on simpler networks of linear threshold units. The output y of such a unit is given by

$$y = \begin{cases} 1 & \text{if } v > w_0 \\ -1 & \text{otherwise,} \end{cases}$$

where $v = \sum_{j=1}^{n} w_j x_j$ is the *total input*, $\mathbf{w} \in \mathbb{R}^n$ is the weight vector and $w_0 \in \mathbb{R}$ the threshold. Since w_0 can be considered as the weight associated to an additional constant input, it will be often omitted. The inputs of the network are assumed to take either binary or real values.

Feedforward networks may be viewed as a generalization of Boolean circuits in which the usual conjunction, disjunction or negation gates are replaced by more powerful units that can compute any linear threshold function. Figure 4.1 shows two simple examples of feedforward networks. From now on, n denotes the number of input units and m the number of output units. The units belonging to the intermediate layers are called *hidden units*. By convention, the input layer is not taken into account when counting the total

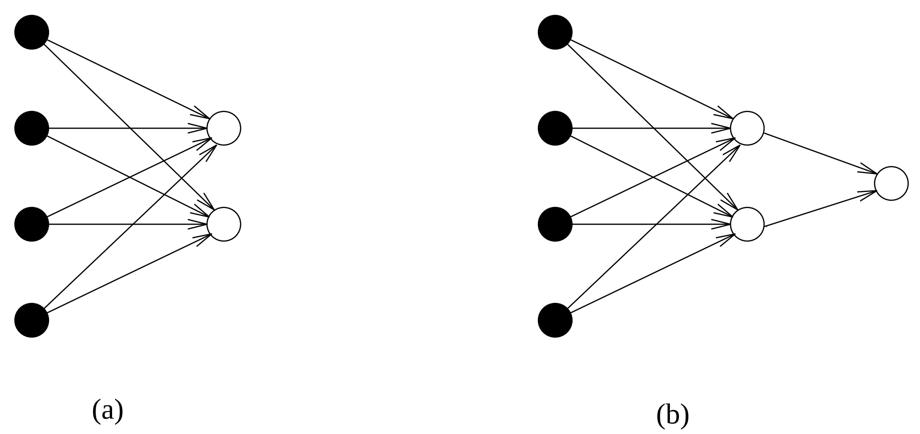


Figure 4.1: Examples of two feedforward networks. Input units are depicted in black. (a) A single-layer network with $n = 4$ input units and $m = 2$ output ones. (b) A two-layer network with two hidden units.

number of layers so that Figure 4.1 (b) illustrates a two-layer network. While input units are simple interfaces, units in the other layers are involved in the information processing. When an input vector $\mathbf{a} \in \mathbb{R}^n$ is presented to the input layer, the outputs of the other units are computed layer by layer, from the second layer to the last one, and the outputs of the last layer units form a vector $\mathbf{b} \in \mathbb{B}^m$ which is the overall output of the network for the given input $\mathbf{a}$. Thus any feedforward network with a fixed set of values for the weights implements an application h from the input space $\mathbb{R}^n$ into the output space $\mathbb{B}^m$. A network with real inputs and a single output unit defines a Boolean-valued function $h : \mathbb{R}^n \to \mathbb{B}$ which is the indicator function of the input space subset $h^{-1}(1)$.

Feedforward networks with continuous or linear threshold units are used to tackle various types of problems ranging from pattern recognition to time series forecasting [152, 50, 164]. Their universal function representation capabilities make them attractive candidates for dealing with such complex problems. If enough hidden layers and hidden units are available, feedforward networks are able to approximate any measurable function within an arbitrary accuracy [117, 51, 90]. In particular, it has been proved that two-layer networks of linear threshold units can approximate arbitrarily well any Boolean-valued function with binary or real inputs, provided a sufficient number of hidden units. When an exact representation is of interest, three-layer networks are needed to define arbitrarily complex regions in the input space [168, 169]. Necessary and sufficient conditions for exact classification with two-layer and three-layer networks have been derived in [169].

Formally, typical problems tackled with feedforward networks are given by a set of p input-output pairs $(\mathbf{a}^k, \mathbf{b}^k) \in \mathbb{R}^n \times \mathbb{B}^m$. In a pattern classification problem, for instance, these pairs correspond to a set of known patterns $\mathbf{a}^k$ with the associated membership class coded as $\mathbf{b}^k$. Such a set of pairs is called the *task* or the *training set* and is denoted by $T = \{(\mathbf{a}^k, \mathbf{b}^k)\}_{1 \le k \le p}$. The purpose is to design a feedforward network that provides an output $\mathbf{o}^k$ as good as possible for each input vector $\mathbf{a}^k$ in the task, i.e. such that $\mathbf{o}^k \approx \mathbf{b}^\mathbf{k}$. In the classical approach we start with a given task, we choose by some rule of thumb a network with a reasonable number of layers and nodes and then we use a *training algorithm* to tune its weights so as to optimize its performances on T. For feedforward networks with continuous units, the famous *backpropagation* algorithm minimizes the least mean squared error using a simple stochastic gradient technique [145]. Unfortunately, there is no guarantee to reach an optimal set of weights because the error function has typically many local minima. In practical network design, appropriate networks are determined by trial and error. Starting with an initial network, the topology (number of layers and hidden units) is modified until the weight values provided by the training algorithm are satisfactory.

In using feedforward networks, the ultimate goal is not only to achieve distributed information storage and retrieval. Since each *association* $(\mathbf{a}^k, \mathbf{b}^k)$ in the task can be viewed as an *example* of an underlying but unknown function $f : \mathbb{R}^n \to \mathbb{B}^m$, i.e. $T = \{(\mathbf{x}^k, f(\mathbf{x}^k))\}_{1 \le k \le p}$, the objective is to find a feedforward network with n input units and m output units implementing a mapping $h : \mathbb{R}^n \to \mathbb{B}^m$ consistent with f on T and which is a good approximation of f on the whole input space. In other words, we look for a network that does not only perform the task satisfactorily but also *generalizes* it. The hope is that a network trained to behave well on a large enough number of examples has extracted the underlying regularities of the task and provides, with some confidence, a reasonable output also for other input vectors that are not in T. This process, commonly referred to as *learning from examples*, can thus be considered as that of approximating an unknown function from a set of sparse data points. In many situations like natural pattern recognition, such a learning paradigm is a very attractive alternative to explicit programming.

In terms of approximation theory, training a given network is searching for a good approximation h of f in the set $\mathcal{H}$ of all functions computable by the network. This is done by tuning the weights so that the function h agrees as well as possible with f on the input vectors in T. Since f is only partially defined by T, h can in principle be any extension of the task. As we will see in the next section, the behavior of the network, for input vectors which do not occur in T, depends on how well the task represents f and on the function representation power of the given network.

In the sequel we shall distinguish between *network training* and *network design* in which not only the weights but also the topology of the network has to be determined. Despite the large resurgence of interest in neural computation, it is still unclear how to design an appropriate network for a given task. This issue will be discussed in chapter 6.

4.2 Computational learning theory

In recent years, the field of computational learning theory which aims at formalizing the study of a variety of learning processes, has undergone a considerable development [134, 17]. In this section we briefly describe the general PAC model of learning from examples that provides a rigorous framework to address two fundamental questions. How many examples must be correctly learned by a network to expect good generalization ? Provided the task conveys enough information, is there an efficient algorithm to train the network ? Our discussion on the hardness of training single-layer networks and of designing close-to-minimum size ones will be based on this framework.

4.2.1 The PAC model

In his seminal paper [159], Valiant proposed a probabilistic definition of learning from examples which formalizes generalization as prediction. Since its introduction, this model and its variants have been gaining acceptance as a fundamental paradigm.

Although multi-valued functions can also be considered [85], the focus here is on learning from examples an unknown Boolean-valued function $f : X \to \mathbb{B}$, called the *target function*. An *example* of f, i.e. a pair $(\mathbf{x}, f(\mathbf{x}))$ for some $\mathbf{x} \in X$, is *positive* or *negative* depending on whether $f(\mathbf{x})$ is equal to 1 or -1. An unknown probability distribution P is assumed to be defined on the input space X. The key assumption is that the training examples as well as the test examples are chosen randomly and independently according to the same but arbitrary distribution P. Besides the examples, some prior information on the target function is also available. Once all functions inconsistent with this information have been eliminated, we are left with a restricted class of potential target functions called the *target space* and denoted by $\mathcal{F}$.

Formally, learning a target function $f \in \mathcal{F}$ from examples consists, given a set of examples, in finding a good approximation in a given set of available functions $\mathcal{H}$, called the *hypothesis space*. For neural networks, the hypothesis space corresponds to the set of all functions computable by the given network. In the basic model, we assume that $\mathcal{F} \subseteq \mathcal{H}$.

The error of a hypothesis h, with respect to the target function f, is taken to be the probability that h disagrees with f on a random example selected according to P. If this probability of error is at most ϵ with $0 < \epsilon < 1$, h is said to be *ε-accurate* for f. A confidence parameter δ, $0 < \delta < 1$, is also introduced to cope with the (unlikely) cases where the training set is not a representative set of random examples of f. In such cases, the learning algorithm is allowed to produce a poor approximation of f with probability at most δ.

Definition 4.2.1 [159] An algorithm is a *probably approximately correct* (PAC) learning algorithm for a target space $\mathcal{F}$ and a hypothesis space $\mathcal{H}$ if, for any $f \in \mathcal{F}$, any probability distribution P and any given accuracy and confidence parameters $0 < \varepsilon, \delta < 1$, it has access to examples of f drawn according to P and produces an ε-accurate hypothesis $h \in \mathcal{H}$ for f with probability greater than $1 - \delta$.

Of course we are interested in efficient PAC-algorithms. The efficiency depends on the *sample complexity*, that is the number of examples needed, as well as on the time complexity. A class of Boolean-valued functions $\mathcal{F}$ is said to be PAC-*learnable* by $\mathcal{H}$ if there exists a PAC-learning algorithm requiring a time complexity and a sample complexity polynomial in $1/\delta$ and $1/\epsilon$. According to the standard terminology, $\mathcal{F}$ is *learnable* if it is PAC-learnable by $\mathcal{H} = \mathcal{F}$.

As will be briefly recalled in sections 4.2.2 and 4.2.3, Valiant's distribution-independent and complexity-based model provides important insights into the problem of learning in feedforward networks. In particular, it allows to investigate the two main issues: sample complexity and computational complexity.

4.2.2 Sample complexity for feedforward networks

In [41] Blumer et al. established upper and lower bounds on the number of examples that a hypothesis must correctly classify in order to have some confidence that it will provide the correct output for future examples drawn from the same distribution. These results involve the important notion of Vapnik-Chervonenkis dimension (*VC dimension*) of a set of Boolean-valued functions.

Let $\mathcal{H}$ be a set of Boolean-valued functions defined on some space X. A finite subset $Y \subseteq X$ is said to be *shattered* by $\mathcal{H}$ if each of the $2^{|Y|}$ possible dichotomies of Y is consistent with at least one Boolean-valued function $h \in \mathcal{H}$.

Definition 4.2.2 The VC dimension of a set of Boolean-valued functions $\mathcal{H}$ is the cardinality of the largest subset of X which is shattered by $\mathcal{H}$, and is denoted by $VCdim(\mathcal{H})$.

This combinatorial quantity measures the "expressive power" of $\mathcal{H}$. For instance, the VC dimension of perceptrons with n real inputs and a threshold is $n+1$ [17].

Blumer et al. [41] have drawn the connection between Vapnik and Chervonenkis' work on the uniform convergence of empirical probability estimates [161] and distribution-independent learning.

Theorem 4.2.1 [41] *Let $\mathcal{H}$ be a set of well-behaved*[1] *Boolean-valued functions defined on $\mathbb{R}^n$, $0 < \epsilon, \delta < 1$ and $0 < \gamma \leq 1$ be a slack parameter. Suppose that p training examples with*

$$p \geq \frac{8}{\gamma^2 \epsilon} \; max\{\ln(\frac{8}{\delta}), 2 \; VCdim(\mathcal{H}) \; \ln(\frac{16}{\gamma^2 \epsilon})\} \tag{4.1}$$

are generated according to a fixed but unknown probability distribution P on X. Then with probability at least $1-\delta$ every function in $\mathcal{H}$ that disagrees with at most a fraction $(1-\gamma)\epsilon$

[1] Some measurability conditions are assumed, see [41].

of the p training examples will provide, with probability at least $1-\epsilon$, the correct output for any input vector drawn according to the same distribution P.

For *consistent* hypotheses, i.e. hypotheses that agree with all training examples, the result holds even if p is at least

$$\frac{4}{\epsilon}\ max\{\log(\frac{2}{\delta}), 2\ VCdim(\mathcal{H})\ \log(\frac{13}{\epsilon})\}. \tag{4.2}$$

Ehrenfeucht et al. [56] established an improved lower bound on the sample complexity for any PAC-learning algorithm.

Theorem 4.2.2 [56] *Let $\mathcal{H}$ be a set of well-behaved Boolean-valued functions of cardinality at least 3 and VC dimension at least 1. Then any* PAC-*learning algorithm guaranteed to produce consistent hypotheses has a sample complexity p strictly greater than*

$$\max\{\frac{VCdim(\mathcal{H})-1}{32\varepsilon}, \frac{1}{\varepsilon}\ \ln(\frac{1}{\delta})\} \tag{4.3}$$

for all $\varepsilon \leq 1/8$ and $\delta \leq 1/100$

In [31] Baum and Haussler derived some bounds on the VC dimension of feedforward networks.

Proposition 4.2.3 [31] *The VC dimension of the class of functions computed by a feedforward network with $N \geq 2$ computing units and W parameters is no greater than $2\ W \log(eN)$, where e is the base of the natural logarithm and W is the total number of weights and thresholds.*

This large upper bound together with (4.2) guarantees that any feedforward network which correctly classifies at least

$$\frac{16W}{\varepsilon}\ \log(eN)\ \log(\frac{13}{\varepsilon})$$

examples can be expected to produce the desired output for future inputs with probability at least $1-\delta$.

Recently Bartlett [29] has shown that the VC dimension of a wide family of feedforward networks with real inputs is at least linear in the number of weights.

Proposition 4.2.4 [29] *The VC dimension of an arbitrary two-layer feedforward network with C connections from the input units to the other units is at least $C+1$.*

A similar result is also valid for some completely connected three-layer networks, i.e. such that all pairs of units in adjacent layers are connected. According to theorem 4.2.2, these bounds imply that the sample complexity for PAC-learning feedforward networks increases at least linearly with the number of weights.

In [45] such worst-case bounds on the sample complexity have been compared to the average generalization performances observed for several tasks. While in some cases a big gap was found, in others both theoretical and experimental bounds turned out to be closely related.

In spite of this discrepancy, both theoretical and experimental results demonstrate that, given a task, a close-to-minimum size network is more likely to achieve valid generalization. The size of a network is defined to be either the number of computing units or the number of free parameters (nonzero weights). Section 4.4 and chapter 6 will be devoted to the fundamental problem of designing near-optimum networks.

4.2.3 PAC-learnability and the hardness of the consistency problem

When studying the computational complexity of learning algorithms, the target and hypothesis spaces are supposed to be *parameterized* by the example size, that is $\mathcal{F} = \cup_{n\geq 1}\mathcal{F}_n$ and $\mathcal{H} = \cup_{n\geq 1}\mathcal{H}_n$ where $\mathcal{F}_n$ and $\mathcal{H}_n$ are spaces defined on examples of size n. Algorithms for learning $\mathcal{F}$ by $\mathcal{H}$ are then required to produce an $h \in \mathcal{H}_n$ for any $f \in \mathcal{F}_n$. In the case of feedforward networks, n is the number of input units.

The PAC-learnability of a target space $\mathcal{F}$ by a hypothesis space $\mathcal{H}$ with $\mathcal{F} \subseteq \mathcal{H}$ turns out to be closely related to the combinatorial problem $\mathcal{H}$-Consistency which has the following decision version:

> Given a set of p positive and negative training examples of size n, does there exist a hypothesis $h \in \mathcal{H}_n$ consistent with all p examples?

Finding a consistent hypothesis is obviously at least as hard as answering the above question.

If $\mathcal{H} = \cup_{n\geq 1}\mathcal{H}_n$ with $VCdim(\mathcal{H}_n)$ polynomial in n and there exists an algorithm that is guaranteed to produce a consistent hypothesis for any training set of size p in polynomial time in p and n, then this consistency algorithm PAC-learns $\mathcal{H}$ by $\mathcal{H}$. Thus, in order to PAC-learn perceptrons with n inputs and a threshold, it suffices to draw a number of examples prescribed by (4.2) with $VCdim(\mathcal{H}_n) = n + 1$ and to determine an appropriate linear threshold function using a polynomial time algorithm for linear programming.

Conversely, Pitt and Valiant [142] showed that efficient PAC-learning implies the existence of an efficient randomized algorithm for finding consistent hypotheses in $\mathcal{H}$.

Theorem 4.2.5 [142] *If* RP $\neq$ NP *and* $\mathcal{H}$-CONSISTENCY *is* NP*-hard for a hypothesis space* $\mathcal{H}$*, then* $\mathcal{H}$ *is not* PAC*-learnable.*

In [95, 96, 97] Judd investigated the hardness of the consistency problem for feedforward networks. In particular, he proved that deciding whether a given two-layer network with Boolean inputs can perform a given task is NP-complete. Blum and Rivest showed in [40] that this decision problem remains intractable even for a very simple class of two-layer and three-unit networks with n inputs, 2 hidden units and a single output unit as illustrated in Figure 4.2 (a). This result was further extended by Lin and Vitter [115] who established that it is still NP-complete for the simplest kind of two-layer networks shown in Figure 4.2 (b), namely those with n inputs, a single hidden unit and an output unit that is also connected to the n inputs. It is worth noting that, while Blum and Rivest's result may

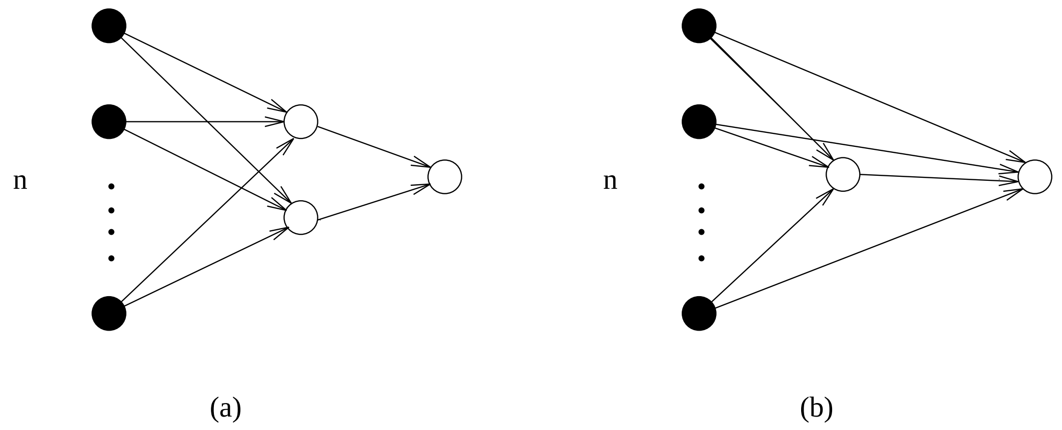


Figure 4.2: Two simple examples of feedforward networks. (a) A two-layer network with n inputs and three computing units. (b) A two-layer network with a single hidden unit, the output unit is connected to the n inputs.

depend on the particular pattern of connection, this is not true for Lin and Vitter's one because any network with two computing units can be obtained from their fully connected network by setting some weights to zero.

Considering approximate training, Judd proved that it is even NP-complete to decide whether a given network can perform correctly more than two-thirds of the associations in a given task [97]. Similar hardness results have also been obtained for feedforward networks whose units compute certain kinds of continuous functions [97, 88].

Since finding a set of weights performing as well as possible a given task is at least as hard as the consistency question, training networks with as few as a single hidden unit is NP-hard. This implies that for any training algorithm there exists some types of tasks and networks for which the amount of time required grows faster than polynomially in the sizes of the task and network. These intractability results accounts for the considerable difficulties encountered when training feedforward networks. For instance, the backpropagation algorithms as well as its variants have no guarantee to reach a global minimum and run very slowly even for networks of moderate size [158].

As previously seen, single linear threshold units (perceptrons) are PAC-learnable. Since the consistency problems are not easier than their decision versions, $\mathcal{H}$-CONSISTENCY is NP-hard for the above-mentioned classes of feedforward networks. According to theorem 4.2.5, the intractability of the corresponding consistency problems implies that these simple types of feedforward networks are not PAC-learnable unless RP = NP. Although this worst-case analysis rules out the existence of efficient learning algorithms working for any task and any network, some particular learning problems might be efficiently solvable.

Over the last years a number of extensions have been proposed to make the basic PAC-learning model more realistic. This includes, for instance, distribution-dependent models, stochastic functions learning and the notion of dynamic sampling (see [17, 116]).

In the next section we shall see that in a more realistic setting the sharp difference between the hardness of learning in feedforward networks with or without hidden units fades out.

4.3 The complexity of training single-layer networks

Although our discussion pertains to single-layer feedforward networks with multiple outputs, we will focus, without loss of generality, on single perceptrons. Indeed, since the m output units of such a feedforward network are independent, each of them can be trained separately. If $T = \{(\mathbf{a}^k, \mathbf{b}^k)\}_{1\leq k\leq p}$ is the overall task, the i-th output unit has to perform the subtask $T_i = \{(\mathbf{a}^k, b_i^k)\}_{1\leq k\leq p}$. Thus the difficulty of training the overall network is equivalent to that of training each single linear threshold unit.

4.3.1 Training perceptrons is harder than linear programming

Before we started our work, a widely accepted “folk” theorem stated that training single-layer feedforward networks is easy while it becomes intractable as soon as there are hidden units. This belief was fostered, among others, by Lin and Vitter's result showing that it is NP-complete to decide whether the simplest type of two-layer network is consistent with a given task [115]. In fact, claims like

> *This result demonstrates a surprising gap between the computational complexities of one-node [perceptron] and two-node neural net training problems* ([115] page 211).

derive from a too restrictive and unrealistic definition of training. Although our discussion goes beyond neural networks, in this section we focus on the particular case of perceptrons.

For any linearly separable task $T = \{(\mathbf{a}^k, b^k)\}_{1\leq k\leq p} \subseteq \mathbb{Q}^n \times \mathbb{B}$, training a perceptron with n inputs and n weights w_j, $1 \leq j \leq n$, amounts to solve the corresponding linear system

$$\begin{array}{l} \mathbf{a}^k\,\mathbf{w} > 0 \;\; \forall k \text{ such that } \; b^k = 1 \\ \mathbf{a}^k\,\mathbf{w} \leq 0 \;\; \forall k \text{ such that } \; b^k = -1, \end{array} \tag{4.4}$$

where the threshold w_0 is considered as an additional weight whose corresponding input always set to -1. Since the consistency problem, named $\mathcal{P}$-CONSISTENCY, is polynomially solvable using linear programming [17], the class of perceptrons is PAC-learnable.

However, the training (learning) problem cannot be satisfactorily casted into that of determining a solution of (4.4) when it exists. Indeed, as mentioned in section 1.2.6, binary tasks of even moderate size have an extremely small probability to be linearly separable. Thus for typical tasks the best we can ask for is a weight vector $\mathbf{w}$ that classifies the p input vectors $\mathbf{a}^k$ as well as possible. Since the focus is on the correctly classified $\mathbf{a}^k$, the natural objective is to look for an *optimal* weight vector that maximizes the number of correct classifications or minimizes the number of misclassifications on T. These complementary problems will be referred to as MAX $\mathcal{P}$-CONSISTENCY and MIN $\mathcal{P}$-DISAGREEMENT respectively, where $\mathcal{P}$ stands for perceptrons. Although they are equivalent to solve optimally, we distinguish them because they turn out to have different approximability properties. MAX $\mathcal{P}$-CONSISTENCY and MIN $\mathcal{P}$-DISAGREEMENT are of course special cases of MAX FLS and MIN ULR with mixed strict and nonstrict inequalities. As we shall see in chapter 6, finding such optimal weight vectors is also a central issue in various constructive strategies for designing multilayer networks of linear threshold units.

In [6] we first pointed out that the difficulty of training perceptrons is not determined by the hardness of the corresponding consistency question. Since training a network is at least as hard as deciding whether it can be consistent with a given task, the NP-completeness of the consistency question directly implies the intractability of training and the non-learnability in the PAC sense. This is the case for the feedforward network classes mentioned in section 4.2.3. However, knowing that the consistency question can be answered in polynomial time (like for perceptrons) tells us nothing about the hardness of the corresponding training problem, except that it is not easier. In such cases, the central issue is how well one can learn a given task when the representation power of the model is overstrained, i.e. when we try to learn a task that is not perfectly performable by a perceptron. From this point of view, the complementary objectives of MAX $\mathcal{P}$-CONSISTENCY and MIN $\mathcal{P}$-DISAGREEMENT are of great interest.

In [6] we established the following result.

Theorem 4.3.1 MAX $\mathcal{P}$-CONSISTENCY *and* MIN $\mathcal{P}$-DISAGREEMENT *are* NP-*hard even when restricted to perceptrons with bipolar inputs and a zero threshold.*

The proof given in [6] extends Johnson and Preparata's NP-hardness result for OPEN HEMISPHERE with coefficients in $\{-4, -2, -1, 0, 1, 4\}$ to the case with strict and nonstrict inequalities as well as bipolar coefficients. This is done using the duplication technique described in the proof of corollary 3.1.3 and enforcing the auxiliary homogeneous equations with pairs of opposite nonstrict inequalities.

Remark 4.3.1 The NP-hardness of the general case with arbitrary threshold and ternary inputs in $\{-1, 0, 1\}$ is an immediate consequence of theorem 3.1.4. Given any MAX FLS$^{>}$ instance with p homogeneous inequalities in n variables and bipolar coefficients, each relation $\mathbf{ax} > 0$ is replaced by $\mathbf{aw} > w_0$ and $p+1$ identical negative examples $\mathbf{0}$ are added to ensure that $w_0 \geq 0$. This new inhomogeneous system is clearly equivalent to the original homogeneous one. Furthermore, there exists a simple trick to make sure that $w_0 \geq 0$ using negative examples that are not identical. The technique will be described in the proof of theorem 4.3.6.

The consistency questions associated to these optimization problems are of course NP-complete.

The next result indicates that the consistency problem does not become easier when the perceptrons can only perform a small subset of linear threshold functions with Boolean inputs.

Theorem 4.3.2 MAX $\mathcal{P}$-CONSISTENCY *and* MIN $\mathcal{P}$-DISAGREEMENT *are* NP-*hard even for perceptrons computing only monomials, i.e. conjunctions of a subset of literals*[2]*.*

PROOF The corresponding decision versions are proved to be NP-complete by polynomial reduction from MIN NODE COVER or equivalently from MAX IND SET. We first show that it is NP-hard to find a monomial that correctly classifies all negative examples and as many positive examples as possible. The basic observation is that there exists a one-to-one correspondence between monomials with at most n literals and the hyperfaces of the n-dimensional hypercube $\mathbb{B}^n$.

Let $G = (V, E)$ be an arbitrary instance of MAX IND SET with $|V| = n$. For each edge $[v_i, v_j] \in E$ we construct a negative example $\mathbf{a} \in \mathbb{R}^n$ such that $a_i = a_j = 1$ and $a_l = -1$ for $1 \leq l \leq n$ with $l \neq i$ and $l \neq j$. For each node $v_i \in V$ we consider the positive example $\mathbf{a} \in \mathbb{R}^n$ such that $a_i = 1$ and $a_l = -1$ for all $l \neq i$. Thus we have a task with $|E| + |V|$ examples.

We claim that there exists an independent set $I \subseteq V$ of size at least s in G if and only if there exists a monomial classifying correctly all negative examples and at least s positive ones.

Given an independent set I of size s in G, consider the monomial in which all the variables corresponding to the nodes in $V - I$ occur negatively. This monomial is true for the s positive examples associated to the nodes in I and false for all the other examples because each of them has at least one variable of $V - I$ equal to 1. Indeed I is an independent set and no edge of G has both extremities in I.

Conversely, suppose we have a monomial that correctly classifies all negative examples and at least s positive ones. Let $I \subseteq V$ denote the set of nodes associated to those positive examples. Since each positive example has its unique 1 in a different position, the variables corresponding to I cannot occur in the monomial. We easily verify that I is an independent set of cardinality s. If I contains two adjacent nodes v_i and v_j, the negative example associated to the edge $[v_i, v_j]$ has all components corresponding to $V - I$ equal to -1. Therefore the monomial cannot distinguish it from the s positive examples that are correctly classified.

The reduction can be easily extended to the unconstrained case with no mandatory examples. It suffices to add enough copies of each negative example to make them more valuable than all positive ones. □

[2]A literal is a Boolean variable or its negation.

Notice that, unlike for linear threshold functions, the consistency problem for monomials is solvable in polynomial time [159].

Our discussion about the relationship between the trainability (learnability) and the two complementary optimization problems associated to the consistency question goes far beyond artificial neural networks. It supports a more realistic model of learning from examples in which there might be no hypothesis consistent with all training examples. As we shall see in the next section, the maximum consistency and minimum disagreement problems will take over the central role played by $\mathcal{H}$-Consistency in the basic PAC framework.

4.3.2 A more realistic PAO-learning model

In the PAC model variant referred to as *approximating from examples* [116], the unrealistic assumption requiring that the target function $f \in \mathcal{F}$ belongs to the hypothesis space $\mathcal{H}$ is relaxed. This strong assumption is not only inappropriate for simple perceptrons. For multilayer networks, it means that the topology chosen by some rule of thumb (guessed) must be able to compute the target function f underlying the task. Now f is unknown and the hypothesis space $\mathcal{H}$ is defined implicitly so that it cannot be easily characterized from the network topology. Furthermore, since smaller networks are more likely to exhibit valid generalization, when guessing the topology one tends to select a network that is too small to perform the given task correctly. Therefore, it is very important in practice that the learning algorithm performances do not degrade abruptly when the representation power of the network is overstrained.

In the general setting, it is assumed that $\mathcal{H}$ contains a hypothesis h which is a reasonable approximation of the target f. Following the terminology in [89], a hypothesis h is said to be ε-*optimal* for f with respect to $\mathcal{H}$ if

$$P\{h\Delta f\} \leq opt(\mathcal{H}) + \varepsilon \quad \text{where} \quad opt(\mathcal{H}) = \inf_{h\in\mathcal{H}} P\{h\Delta f\}$$

and $P\{h\Delta f\}$ is the probability that f and h classify a randomly selected example differently.

Definition 4.3.1 [89] An algorithm is a *probably approximately optimal* (PAO) learning algorithm for $\mathcal{F}$ and $\mathcal{H}$ if, for any $f \in \mathcal{F}$, any probability distribution P and any given accuracy and confidence parameters $0 < \varepsilon, \delta < 1$, it has access to examples of f drawn according to P and produces an ε-optimal hypothesis $h \in \mathcal{H}$ for f with probability greater than $1 - \delta$.

For parameterized spaces, we are interested in efficient algorithms with sample and time complexity polynomial in n as well as in $1/\delta$ and $1/\varepsilon$. $\mathcal{F}$ is PAO-*learnable* by $\mathcal{H}$ if such a PAO-learning algorithm exists.

Of course, if $\mathcal{F} \subseteq \mathcal{H}$ ε-optimal is equivalent to ε-accurate and PAO-learning reduces to PAC-learning. In general, we have $\mathcal{H} \subset \mathcal{F}$. The extreme situation where $\mathcal{F} = \mathcal{A}$ (the set of all well-behaved Boolean-valued functions defined on the input space X) is often of interest. $\mathcal{H}$ is said to be *robustly learnable* if $\mathcal{A}$ is PAO-learnable by $\mathcal{H}$ [1].

Not surprisingly, the PAO-learnability of $\mathcal{H}$ is closely related to the difficulty of the optimization problems associated to $\mathcal{H}$-Consistency. In [89] the connection to the minimization disagreement question was pointed out.

> Min $\mathcal{H}$-Disagreement question: Given a set of p positive and negative examples of size n and an integer K with $1 \leq K \leq p$, does there exist a hypothesis $h \in \mathcal{H}_n$ which misclassifies at most K examples ?

A randomly drawn training set is $\mathcal{F}$-*legal* if $\mathcal{F}$ contains a target function f which is consistent with it.

Theorem 4.3.3 [1] *If* RP $\neq$ NP *and* Min $\mathcal{H}$-Disagreement *restricted to $\mathcal{F}$-legal training sets is* NP-*hard, then $\mathcal{F}$ is not* PAO-*learnable by $\mathcal{H}$.*

Thus, according to theorem 4.3.1, the class of perceptrons with even ternary inputs is not robustly learnable.

As we mentioned in the previous section, Min $\mathcal{H}$-Disagreement is not the only natural optimization problem associated to $\mathcal{H}$-Consistency. By focusing on the number of correctly classified examples, we obtain the complementary maximum consistency problem which may have a different approximability.

> Max $\mathcal{H}$-Consistency question: Given a set of p positive and negative examples of size n and an integer K with $1 \leq K \leq p$, does there exist a hypothesis $h \in \mathcal{H}_n$ which correctly classifies at least K examples ?

The above PAO-learnability result is also valid for Max $\mathcal{H}$-Consistency.

Corollary 4.3.4 *If* RP $\neq$ NP *and* Max $\mathcal{H}$-Consistency *restricted to $\mathcal{F}$-legal training sets is* NP-*hard, then $\mathcal{F}$ is not* PAO-*learnable by $\mathcal{H}$.*

Proof Since Max $\mathcal{H}$-Consistency and Min $\mathcal{H}$-Disagreement are complementary problems, they are equivalent to solve optimally and we can simply transpose the proof of theorem 4.3.3.

We just show that any PAO-learning algorithm for $\mathcal{H}$ would lead to a Monte Carlo algorithm for the corresponding Max $\mathcal{H}$-Consistency. Suppose that the input space X on which are defined the target functions and the hypotheses is discrete. Given a set of p examples (each one being either positive or negative) and a bound K with $1 \leq K \leq p$, the randomized algorithm proceeds as follows. Take $\varepsilon = 1/(p+1)$ and $\delta = 1/2$. Let f be any function of $\mathcal{F}$ consistent with the p examples and the probability distribution P be defined by $P(\mathbf{x}) = \mathrm{nb}(\mathbf{x})/p$, where $\mathrm{nb}(\mathbf{x})$ is the number of occurences of $\mathbf{x}$ in the training set. By applying the PAO-learning algorithm with the above choice of ε, δ, f and P we obtain a hypothesis $h \in \mathcal{H}$. Then we give a positive answer to the corresponding Max $\mathcal{H}$-Consistency question if and only if h correctly classifies at least K examples of the training set.

This procedure is easily verified to be a Monte Carlo algorithm. If no hypothesis in $\mathcal{H}$ correctly classifies at least K examples, the answer is obviously negative. Conversely, if such a hypothesis exists a PAO-learning algorithm is guaranteed to produce with probability at most $1-\delta = 1/2$ an h that correctly classifies at least $\lceil K - p \cdot \varepsilon \rceil = K$ examples. Whenever this happens the Max $\mathcal{H}$-Consistency question is answered positively.

The cases with a continuous input space X can be dealt with similarly. $\square$

Although, assuming RP $\neq$ NP, the NP-hardness of Max $\mathcal{P}$-Consistency and Min $\mathcal{P}$-Disagreement rules out the robust learnability of perceptrons, we shall see that their approximability plays an important role in a slightly relaxed and more natural setting. While we mainly studied the approximability of Max $\mathcal{P}$-Consistency, others focused on that of Min $\mathcal{P}$-Disagreement. In section 4.3.5 we will explain why the approximability of Max $\mathcal{P}$-Consistency is more relevant to the hardness of training perceptrons.

4.3.3 Approximability of Min $\mathcal{P}$-Disagreement

In [89] Höffgen, Simon and van Horn proved that Min $\mathcal{P}$-Disagreement is at least as hard to approximate as Set Covering. Since their result holds for perceptrons with binary inputs in $\{0,1\}$ and an arbitrary threshold, it slightly strengthens our NP-hardness result for ternary inputs. They also show that Min $\mathcal{P}$-Disagreement restricted to perceptrons computing monomials is still Min Dominating Set-hard and that it becomes Min PB-complete (hence NPO PB-complete) for perceptrons with binary weights.

It is worth noting that the approximability status of Set Covering was not set at that time. Set Covering was known to have no Ptas and an $O(\log n)$-approximation algorithm, where n is the size of the groundset.

Perceptrons being not robustly learnable unless RP = NP, Höffgen et al. suggested a natural relaxation of the robust learning definition. Instead of requiring a hypothesis whose prediction error is away from the optimum by a constant $\varepsilon > 0$, one looks for a hypothesis that approximates it within a constant factor. Moreover, the model is modified as in *agnostic learning* where the assumptions on the target function are weakened [105]. In order to try to overcome the intractability of finding a hypothesis with minimal error in a given space $\mathcal{H}'$, a larger hypothesis space $\mathcal{H} \supseteq \mathcal{H}'$ is considered but success is still measured with respect to $opt(\mathcal{H}')$. In some cases, this relaxed setting allows to circumvent certain representational problems due to the particular choice $\mathcal{H}' = \mathcal{H}$ [105].

Definition 4.3.2 [89] The $\mathcal{H}'$-degradation of a learning algorithm for $\mathcal{F}$ and $\mathcal{H}$ is *limited by d* if, in the confident case, it produces a hypothesis $h \in \mathcal{H}$ with a prediction error at most $d \cdot opt(\mathcal{H}') + \varepsilon$. The $\mathcal{H}'$-degradation of $\mathcal{H}$ is *unlimited* when learning target functions in $\mathcal{F}$ if the degradation is not limited by any fixed constant $d \geq 1$.

In general, $\mathcal{H}$ and $\mathcal{A}$ are taken as default for $\mathcal{H}'$ and $\mathcal{F}$, so that $\mathcal{H}$ has an *unlimited degradation* if the $\mathcal{H}$-degradation of $\mathcal{H}$ is unlimited when learning targets in $\mathcal{A}$.

Not surprisingly, the fact that the $\mathcal{H}'$-degradation of $\mathcal{H}$ is limited or unlimited depends on the approximability of Min $\mathcal{H}$-Disagreement. A training set is said to be $(\mathcal{H}, \mathcal{H}')$-*indifferent* if the minimum number of misclassifications achievable by hypotheses in $\mathcal{H}'$ is equal to that achievable by those in $\mathcal{H}$.

Theorem 4.3.5 [89] *Let $\mathcal{H}' \subseteq \mathcal{H}$. If* RP $\neq$ NP *and it is* NP*-hard to approximate* Min $\mathcal{H}$-Disagreement *restricted to $\mathcal{F}$-legal and $(\mathcal{H}, \mathcal{H}')$-indifferent training sets to within a factor d, then the $\mathcal{H}'$-degradation of $\mathcal{H}$ is not limited by d when learning targets in $\mathcal{F}$.*

As conjectured in [89], perceptrons and monomials turned out to have unlimited degradation. This is an immediate consequence of [120, 32] where Set Covering was shown not approximable within a constant factor unless P=NP and within $c \log(n)$ for any $c < 1/8$ unless NP $\subseteq$ DTIME$(n^{\log \log n})$.

In [18] Arora, Babai, Stern and Sweedyk suggest a way of extending their strong non-approximability bound for Min ULR$^{=}$ to strengthen the one for Min $\mathcal{P}$-Disagreement with real weights and binary inputs. In fact, they consider the symmetric variant of Min

$\mathcal{P}$-DISAGREEMENT where we ask $\mathbf{a}^k\mathbf{w} > w_0$ for positive examples and $\mathbf{a}^k\mathbf{w} < w_0$ for negative ones. Although their argument does not suffice to complete the proof, it can easily be fixed.

The problem is related to the fact that starting with an arbitrary instance of MIN ULR$^=$ with binary coefficients one must construct a system with strict inequalities with a special variable playing the role of the threshold w_0. As mentioned in [18], any considered instance of MIN ULR$^=$ can be easily expressed as an equivalent inhomogeneous instance of MIN ULR$^>$. Therefore each equation is replaced by two opposite nonstrict inequalities, and a new slack variable δ is added to turn nonstrict inequalities into strict ones. More precisely, every relation $\mathbf{aw} \geq 0$ is replaced by $\mathbf{aw} + \delta > 0$. Now, in order to make sure that the two systems are equivalent we must have $\delta < 1/L$, where L is a given positive integer. This can of course be guaranteed by introducing a large enough number of copies of this strict inequality, but then the resulting system is not an instance of symmetric MIN $\mathcal{P}$-DISAGREEMENT. Indeed, if δ is considered as the threshold the inequalities ensuring $\delta < 1/L$ are inhomogeneous.

Fortunately, there exists a simple and general way of constructing, for any instance of inhomogeneous MIN ULR$^>$, an equivalent instance of symmetric MIN $\mathcal{P}$-DISAGREEMENT.

Remark 4.3.2 Suppose we have a system $\mathbf{a}^k\mathbf{w} > b^k$ with $1 \leq k \leq p$ where all b^k are nonzero. Multiply each inequality by an appropriate constant so that all right hand sides are equal to 1. By replacing all right hand sides constants 1 by a variable w_0, we get a system with either $\mathbf{a}^k\mathbf{w} > w_0$ type or $\mathbf{a}^k\mathbf{w} < w_0$ type inequalities. Clearly, any solution of this new system such that $w_0 > 0$ gives a solution of the original system. Thus by adding a large enough number of copies of $w_0 > 0$ the two problems are guaranteed to be equivalent.

In order to complete the reduction in [18], we just apply this technique to the system consisting of $\mathbf{aw} + \delta > 1/(2L)$ inequalities and enough copies of $\delta < 1/L$. Thus symmetric MIN $\mathcal{P}$-DISAGREEMENT cannot be approximated within $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$ unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\mathrm{polylog}\, n})$. Since in the reduction the number of variables n and relations p are polynomially related, the nonapproximability bound still holds when n is replaced by p.

It should be pointed out that this proof is only valid if degenerate training sets containing many copies of some examples are allowed. The same reduction can of course be extended to standard (nonsymmetric) MIN $\mathcal{P}$-DISAGREEMENT.

4.3.4 Approximability of MAX $\mathcal{P}$-CONSISTENCY

We now turn to the complementary problem of maximizing perceptron consistency. Since it is a special case of MAX FLS with strict and nonstrict inequalities, MAX $\mathcal{P}$-CONSISTENCY can be approximated within a factor of 2 using the greedy algorithm of proposition 3.2.6. Although MAX $\mathcal{P}$-CONSISTENCY is much easier to approximate than MIN $\mathcal{P}$-DISAGREEMENT, it does not admit a polynomial time approximation scheme.

Proposition 4.3.6 MAX $\mathcal{P}$-CONSISTENCY *is* APX-*complete and can be approximated within* 2.

PROOF In order to show that the problem is MAX SNP-hard, and hence APX-hard, we adapt the L-reduction from MAX 2SAT to MAX FLS$^>$ given in the proof of theorem 3.2.2. Starting with the system of $10m$ strict inequalities (3.30)–(3.39), we add an extra variable w_0 (the threshold) and a large enough number of nonstrict inequalities forcing $w_0 \geq 0$. For each clause C_i, $1 \leq i \leq m$, containing two variables x_{j_1} and x_{j_2} we consider the 10 n-dimensional positive examples with $n = |X|+1$ corresponding to the following inequalities:

$$\begin{aligned}
10a_{ij_1}w_{j_1} + 10a_{ij_2}w_{j_2} + 2w_{|X|+1} &> w_0 && (4.5)\\
10w_{j_1} - 8w_{|X|+1} &> w_0 && (4.6)\\
-10w_{j_1} + 12w_{|X|+1} &> w_0 && (4.7)\\
10w_{j_1} + 12w_{|X|+1} &> w_0 && (4.8)\\
-10w_{j_1} - 8w_{|X|+1} &> w_0 && (4.9)\\
10w_{j_2} - 8w_{|X|+1} &> w_0 && (4.10)\\
-10w_{j_2} + 12w_{|X|+1} &> w_0 && (4.11)\\
10w_{j_2} + 12w_{|X|+1} &> w_0 && (4.12)\\
-10w_{j_2} - 8w_{|X|+1} &> w_0 && (4.13)\\
w_{|X|+1} &> w_0. && (4.14)
\end{aligned}$$

Moreover, we include $3m$ identical negative examples $\mathbf{0}$ implying the inequality $w_0 \geq 0$. Thus we have a MAX $\mathcal{P}$-CONSISTENCY instance with $10m$ positive examples and $3m$ negative ones.

It is easy to verify that there is a correspondence between solutions of the MAX 2SAT instance satisfying s clauses and perceptrons classifying correctly $10m + s$ vectors of the task. In fact, in any solution $(\mathbf{w}, w_0)$ which fulfills $10m + s$ inequalities we have $w_{|X|+1} > w_0 \geq 0$ because the corresponding hyperplane must correctly classify at least a negative example and a positive one of type (4.14). As far as the L-reduction is concerned, the inequalities (4.5)–(4.14) are therefore equivalent to the inequalities (3.30)–(3.39).

This L-reduction can be extended to the case where the negative examples are not all identical. This can be achieved by considering $n + 3m$–dimensional examples instead of n–dimensional ones. We construct the same $10m$ positive examples (all additional components are 0) and $3m$ pairs of negative examples implying the following inequalities:

$$\begin{aligned} w_{|X|+i+1} &\leq w_0 \\ -w_{|X|+i+1} &\leq w_0 \end{aligned}$$

where $1 \leq i \leq 3m$ and $w_{|X|+i+1}$ are (free) variables occurring in a single inequality. Clearly, any solution $(\mathbf{w}, w_0)$ satisfying at least one pair of inequalities is such that $w_0 \geq 0$. Since the absolute values of all $w_{|X|+i+1}$ can be taken as small as needed (these variables are unconstrained) and since at most $6m$ inequalities of types (3.31)–(3.38) can be simultaneously satisfied, we have a correspondence between solutions of the Max 2Sat instance satisfying s clauses and perceptrons correctly classifying $13m + s$ examples. □

The same nonapproximability result holds for symmetric Max $\mathcal{P}$-Consistency. One easily verifies that the problem is also approximable within 2 when the perceptrons are restricted to perform monomials. As for Max FLS$^{\geq}$ and Max FLS$^{>}$, the question of whether a better constant factor can be guaranteed in polynomial time is still open.

It is interesting to note that Max $\mathcal{P}$-Consistency remains Apx-complete even for perceptrons with binary weights. Therefore it is much easier to approximate than the corresponding version of Min $\mathcal{P}$-Disagreement which is NPO PB-complete.

Proposition 4.3.7 Max $\mathcal{P}$-Consistency *restricted to perceptrons with binary weights, binary inputs and an integer threshold is* Apx*-complete and can be approximated within* 2.

Proof As for perceptrons with real weights, it is always possible to correctly classify at least half of the examples in any binary task. If the number of positive examples $|T^+|$ is greater than the number of negative examples $|T^-|$, one takes all weights $w_j = 1$ and the threshold $w_0 = -1$. Such a perceptron classifies all examples as positive and makes at most $\lfloor p/2 \rfloor$ misclassifications. Conversely, if $|T^-| > |T^+|$ the perceptron with $\mathbf{w} = \mathbf{0}$ and $w_0 = n + 1$ classifies all examples as negative.

To prove Apx-hardness, we proceed by L-reduction from Max 2Sat. Let $(X, \mathcal{C})$ with $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an arbitrary instance of Max 2Sat. For each clause C_i, $1 \leq i \leq m$, containing two variables x_{j_1} and x_{j_2} we construct three examples (one positive and two negatives) with $n = 2|X| + 2$ components. The first $|X|$ components correspond to the Boolean variables x_j, $1 \leq j \leq |X|$, the next $|X|$ ones to their negations $\bar{x}_j$ and the two last ones are used, in particular, to force $w_0 = 1$. The positive example $\mathbf{a} \in \{0, 1\}^n$ is defined as follows. We set $a_{j_k} = 1$ or $a_{|X|+j_k} = 1$ depending on whether x_{j_k} occurs positively or negatively in the clause. Furthermore, we take $a_{2|X|+1} = 1$ and all other components equal to zero. The two negative examples correspond to the nonstrict inequalities

$$\begin{aligned} w_{j_1} + w_{|X|+j_1} &\leq w_0 \\ w_{j_2} + w_{|X|+j_2} &\leq w_0. \end{aligned} \tag{4.15}$$

Their role is to make sure that, for $k = 1, 2$, w_{j_k} and $w_{|X|+j_k}$ associated to the Boolean variable x_{j_k} and its negation are not simultaneously equal to 1. Finally, we add m copies of the negative example corresponding to the inequality

$$w_{2|X|+1} \leq w_0 \tag{4.16}$$

and m copies of the positive example $\mathbf{a}$ with corresponding to

$$w_{2|X|+1} + w_{2|X|+2} > w_0 \tag{4.17}$$

which forces $w_0 < 2$. Thus we have a task with $5m$ examples with $n = 2|X|+2$ components.

Given a truth assignment that satisfies s clauses of the MAX 2SAT instance, we can easily construct a perceptron with binary weights that correctly classifies $4m+s$ examples. This is achieved by setting $w_j = 1$, $w_{|X|+j} = 0$ if x_j is TRUE in the assignment, $w_j = 0$, $w_{|X|+j} = 1$ if it FALSE, $w_{2|X|+1} = w_{2|X|+2} = 1$ and $w_0 = 1$. It is readily verified that such a perceptron correctly classifies all negative examples, the m positive examples of type (4.17) and s positive examples associated to the clauses.

Conversely, consider any binary perceptron $(\mathbf{w}, w_0)$ that correctly classifies $4m + s$ examples of the task. Since at least one example of types (4.16) and (4.17) are correctly classified, we must have $0 \leq w_0 \leq 1$. In fact, $w_0 = 1$ because $w_0 = 0$ would imply that $w_{2|X|+1} = 0$ and hence that no more than $4m$ examples can be correctly classified (at most two out of the three associated to each clause). Given that $w_0 = 1$, if a pair of weights w_{j_k} and $w_{|X|+j_k}$ corresponding to a Boolean variable x_{j_k} are both 0, then one of them can be set to 1 without decreasing the number of correctly classified examples. In other words, we can assume that the binary weight vector $\mathbf{w}$ is such that for all k, $1 \leq k \leq |X|$, w_{j_k} and $w_{|X|+j_k}$ have opposite binary values. Thus the truth assignment given by

$$x_j = \begin{cases} \text{TRUE} & \text{if } w_j = 1 \text{ and } w_{|X|+j} = 0 \\ \text{FALSE} & \text{otherwise,} \end{cases}$$

for $1 \leq j \leq |X|$, satisfies s clauses of the original MAX 2SAT instance.

Consequently, we have a correspondence between solutions of the MAX 2SAT instance satisfying s clauses and perceptrons with binary weights that correctly classify $4m + s$ examples. Since opt[MAX $\mathcal{P}$-CONSISTENCY] $\leq 5m$ and since there exists an algorithm providing a truth assignment that satisfies at least $\lceil \frac{m}{2} \rceil$ of the clauses in any MAX 2SAT instance, we have opt[MAX $\mathcal{P}$-CONSISTENCY] $\leq 10 \cdot$ opt[MAX 2SAT]. Therefore all conditions for an L-reduction are fulfilled.

Using additional components, one can avoid having m identical examples of types (4.16) and (4.17). □

As for the real weights, there might exists a better polynomial time approximation algorithm.

If ternary inputs are allowed, MAX $\mathcal{P}$-CONSISTENCY with binary weights and a positive threshold becomes MAX IND SET-hard. This can be shown by adapting the cost preserving transformation used in the proof of theorem 3.2.13.

Notice that, unlike for perceptrons with real weights, for binary weights even the consistency question is NP-hard [142]. Recently Schmitt investigated the border line between intractability and tractability of MAX $\mathcal{P}$-CONSISTENCY with binary weights depending on some task characteristics [150].

Finally, we consider a variant of MAX $\mathcal{P}$-CONSISTENCY with real weights that occurs as a subproblem in various constructive methods for designing multilayer networks [124, 123]. In this special case of C MAX FLS$^{\geq;>}$, named MAX $\mathcal{P}$-COVERING, only a single type of misclassifications is allowed. Given a set of examples, we look for a weight vector that correctly classifies all negative examples and as many positive ones as possible.

Proposition 4.3.8 MAX $\mathcal{P}$-COVERING *is* MAX IND SET*-hard.*

PROOF By L-reduction from MAX 2 ONES NEG which is known to be MAX IND SET-hard [136] and is defined as follows [66]. Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of 2-literal clauses with only negated variables, find a truth assignment for X that satisfies every clause and that contains as many TRUE variables as possible.

For each clause C_i, $1 \leq i \leq m$, we construct a negative example $\mathbf{a} \in \mathbb{R}^n$ with $n = |X|$. If a single Boolean variable x_j occurs in C_i, we consider the negative example $\mathbf{a}$ where $a_j = 1$ and $a_l = 0$ for $1 \leq l \leq n$ with $l \neq j$. If two variables x_{j_1} and x_{j_2} occur in C_i, we consider the negative example $\mathbf{a}$ where $a_{j_1} = a_{j_2} = 1$ and $a_l = 0$ for $1 \leq l \leq n$ with

$l \neq j_1$ and $l \neq j_2$. The trivial vector $\mathbf{0}$ is also included as a negative example. Finally, we construct for each Boolean variable x_j the positive example $\mathbf{a}$ where $a_j = 1$ and $a_l = 0$ for $1 \leq l \leq n$ with $l \neq j$.

We claim that there exists a truth assignment with s TRUE variables satisfying all clauses if and only if there exists a perceptron classifying correctly all negative examples and s positive ones. Given an appropriate truth assignment, the hyperplane specified by the threshold $w_0 = 1$ and the normal vector $\mathbf{w}$ defined by

$$w_j = \begin{cases} 2 & \text{if } x_j \text{ is TRUE} \\ -2 & \text{otherwise} \end{cases}$$

correctly classifies all negative examples and the positive ones corresponding to a TRUE variable. Moreover, the $n - s$ positive examples associated with a FALSE variable are misclassified. Conversely, given an appropriate perceptron with a weight vector $\mathbf{w}$ and a threshold w_0 we consider the set of Boolean variables $Y \subseteq X$ associated to a positive example that is correctly classified. It is easily verified that the assignment where the s variables in Y are TRUE and all the other ones are FALSE satisfies all the clauses of $(X, \mathcal{C})$. □

The variant of MAX $\mathcal{P}$-COVERING where the objective is to correctly classify all positive examples and as many negative examples as possible is also MAX IND SET-hard.

4.3.5 The true difficulty of training single-layer networks

As we have seen in sections 4.3.3 and 4.3.4, the approximability of MAX $\mathcal{P}$-CONSISTENCY differs enormously from that of MIN $\mathcal{P}$-DISAGREEMENT even when the weights are restricted to take binary values. Since these complementary objectives provide two opposite views of the same computational problem, the question arises as to which one is more relevant to the problem of training perceptrons. Unlike in many optimization problems, such as the traveling salesman, the appropriate objective function is not clearly defined in this case. A priori, both the number of correctly classified examples and the number of misclassifications are good candidates to measure the network's performance on the training set. It is mainly a matter of bias: optimistic users would count correct classifications while pessimistic ones would focus on misclassifications. Thus both complementary optimization problems associated to the consistency question need to be investigated.

From the learning process point of view, the enormous difference between the hardness

	MAX $\mathcal{P}$-CONSISTENCY	MIN $\mathcal{P}$-DISAGREEMENT
Real weights	APX-complete (within 2)	not within $2^{\log^{1-\varepsilon} s}$ for any $\varepsilon > 0$ [18]
Binary weights		NPO PB-complete [89]

Table 4.1: Approximability bounds for the two complementary optimization versions of the perceptron consistency problem. Both cases with real and binary weights are mentioned. The bound for MIN $\mathcal{P}$-DISAGREEMENT with real weights holds under the usual polylogarithmic assumption where s can either be the number of variables n or relations p.

of approximating MIN $\mathcal{P}$-DISAGREEMENT and MAX $\mathcal{P}$-CONSISTENCY can be considered as an artifact due to the choice of the objective function. Table 4.1 summarizes the known results for perceptrons with real and binary weights. Notice that for real weights these bounds hold even if the inputs are restricted to take integer values.

Since complexity theory is already based on a worst-case analysis, the easiest optimization problem associated to the consistency question provides a more realistic bound on the inherent difficulty of the training problem.

As far as the learning model is concerned, it is just a matter of how we decide to measure performance. Instead of considering the misclassification probability, we can focus on the complementary probability of correct classification. A hypothesis h is then said to be *ε-optimal* for f with respect to $\mathcal{H}$ if

$$P\{h = f\} \geq opt(\mathcal{H}) - \varepsilon \qquad \text{where} \quad opt(\mathcal{H}) = \sup_{h \in \mathcal{H}} P\{h = f\}$$

and $P\{h = f\}$ is the probability that f and h classify a randomly selected example identically. Given this definition, all above-mentioned learnability results can be easily transposed. In particular, the analog of theorem 4.3.5 can be established by proceeding like in the proof of corollary 4.3.4. It suffices to verify that a learning algorithm for $\mathcal{F}$ and $\mathcal{H}$ whose $\mathcal{H}'$-degradation is limited by a constant $d \geq 1$ can be turned into a Monte Carlo algorithm for approximating MAX $\mathcal{P}$-CONSISTENCY restricted to $\mathcal{F}$-legal and $(\mathcal{H}, \mathcal{H}')$-indifferent training sets to within a factor d.

Thus, in our more realistic setting where MAX $\mathcal{P}$-CONSISTENCY plays the central role, we have:

Corollary 4.3.9 *Perceptrons with real or binary weights allow limited degradation but there exists a constant $1 < d < 2$ such that the degradation cannot be limited by d.*

Höffgen et al. discussion was just too pessimistic and their nonapproximability result for Min $\mathcal{P}$-Disagreement with binary weights, as well as Arora's et al. one for real weights, have no real implications on the hardness of the learning problem. In fact, training single perceptrons is harder than linear programming but it is polynomially solvable if hypotheses that approximate the target function within a constant 2 suffice. According to proposition 4.3.8, the problem becomes much harder when only a single type of errors is allowed.

These results provide strong evidence that the performance of any algorithm for training perceptrons degrade up to a given constant factor $1 < d < 2$ when their representation power is overstrained. As previously noted, this is also valid for single-layer networks.

Of course, our discussion about the relevance of Max $\mathcal{H}$-Consistency and Min $\mathcal{H}$-Disagreement to the learning problem goes far beyond artificial neural networks. By studying the particular case of perceptrons, we have shown that the learnability results obtained in computational learning theory strongly depend on the choice of the objective function used to extend $\mathcal{H}$-Consistency. In particular, we proved that the non-learnability results derived from the hardest optimization version of the consistency problem may be much too pessimistic and hence of minor interest.

Our negative results are not only interesting on their own, but also because they suggest how to modify the learning model to make it more realistic. Since for other hypothesis spaces Min $\mathcal{H}$-Disagreement might be easier than Max $\mathcal{H}$-Consistency, the complexity of approximating both optimization versions of $\mathcal{H}$-Consistency must be investigated. If these complementary problems are not equally hard to approximate, the easiest one should be taken into account and the prediction error should be measured accordingly. Although we conjecture that for common hypothesis spaces Min $\mathcal{H}$-Disagreement is at least as hard as Max $\mathcal{H}$-Consistency, we leave as an open problem to establish whether this is always true.

Discriminating between Max $\mathcal{H}$-Consistency and Min $\mathcal{H}$-Disagreement, is thus an important step towards a more realistic model of approximate learning from examples. The choice of the appropriate measure of performance naturally fits into the variations of the basic model that aim at making it more relevant to the many hypothesis spaces with intractable consistency problems (see the references in [17]).

4.4 The hardness of designing close-to-minimum size networks

As seen in section 4.2.2, network design is a major issue in the field of artificial neural networks. Theoretical as well as experimental results show that the size of a network has a strong impact on its generalization properties. According to Occam's principle, among all networks capable of performing a given task, the more compact ones are more likely to exhibit good generalization [134]. The size of a network is given either by the number of nonzero weights or by the number of computing units. While chapter 6 will be devoted to algorithms for designing compact networks, this section is concerned with the hardness of the design problem.

Thus, learning from examples in feedforward networks consists rather in designing a minimum size network performing a given task T than in training a given network to make it consistent with T. Computationally, the topology of the network is no longer part of the input and must be determined during the learning phase. From the learning model perspective, the general design problem amounts to *representation-independent* learning where any polynomial time evaluable hypothesis is considered instead of restricting to a specific space $\mathcal{H}$. This is in contrast with the training problem where the hypothesis space is defined by the topology of the network at hand. Note that representation-independent hardness results are needed in order to establish whether a given target class is efficiently learnable. Indeed, proving that a target space $\mathcal{F}$ is not efficiently learnable by a certain $\mathcal{H}$ does not preclude the possibility that it is efficiently learnable by a different hypothesis space [104].

Designing a minimum size network is of course at least as hard as the consistency question. However, optimum networks are not required in practice and near-optimum ones that have good generalization properties suffice. The central question is how close to the optimum one can get. On one hand, any binary task can be performed by a two-layer network implementing a look-up table [52] but such oversized systems behave poorly on unseen data. On the other hand, several computational complexity results indicate that designing a close-to-minimum size network is also a difficult problem.

The NP-hardness of the consistency question for the simplest kind of two-layer networks [115] has an immediate consequence on the hardness of designing near-optimum networks. To our knowledge, this direct implication of Lin and Vitter's result has not been pointed out in the literature.

Corollary 4.4.1 *Assuming* P $\neq$ NP*, no polynomial time algorithm can design, for any given binary task T, a feedforward network consistent with T and having strictly less than twice as many hidden units as the minimum network.*

Proof First recall that any network with two computing units can be obtained from such fully connected network by setting some weights to zero. Then suppose that such a polynomial time algorithm exists. For any binary task T performable by this type of a network, it would produce a network consistent with T that has strictly less than $1 \cdot 2$, i.e. 1, hidden units. But this is in contradiction with the NP-hardness of the consistency question. □

In other words, assuming P $\neq$ NP, for any polynomial time training algorithm there will be some tasks for which it fails to design a consistent as well as near-optimum network.

In [104] Kearns and Valiant established a much stronger nonapproximability bound but under a stronger cryptographic assumption. In particular, they proved that if trapdoor functions[3] exist it is intractable to find a feedforward network with a bounded number of layers that performs a given task and that is at most polynomially larger than the minimum possible one. The size is here measured in terms of the number of bits needed to describe the network. Note that such a cryptographic assumption is much stronger than P $\neq$ NP because (i) it implies the latter one while the converse is not known to be true and (ii) specific problems are assumed to be hard. This result demonstrates that even approximating minimum networks within polynomial ratios is intractable. However, it leaves open the possibility that this strong nonapproximability bound depends on the fact that intricate networks with a large number of hidden layers may be considered. Indeed, the target functions that Kearns and Valiant proved hard to learn are the very special inverses of trapdoor functions. Of course, the question arises of how relevant such a cryptographic zero measure subclass is to real-world learning.

In order to avoid artifacts due to very complex tasks and topologies, it is interesting to investigate the difficulty of designing close-to-minimum size perceptrons. In this case, only linearly separable tasks are considered and the objective is to minimize the number of nonzero weights required to correctly classify all examples. Although most tasks are nonlinearly separable, this situation naturally occurs when designing compact networks. If

[3] A trapdoor function $\mathcal{T}$ is a one-to-one function such that $\mathcal{T}$ and its inverse are easy to evaluate but, given $\mathcal{T}$, the inverse $\mathcal{T}^{-1}$ cannot be constructed in polynomial time.

the selected network is too small to perform a given task satisfactorily, its representation power can be increased in two different ways. Either one expands its topology by adding new computing units and connections or one modifies the input encoding by providing additional information. As well known, the choice of an appropriate preprocessing has a major impact on the network performances. In some cases, a training problem that is hard for some network can even become tractable for an extended encoding of the input and a more powerful network [40]. In practice, if the n_o original features do not suffice, new inputs can be derived from the original ones. For instance, the $O(n_o^d)$ higher-order products of the original features are often included for several values of $d \geq 2$. Although the resulting number of inputs n is much larger than n_o, only a small fraction of them are frequently needed.

> OPTIMUM PERCEPTRON: Given a binary task containing p examples with n-dimensional input vectors in $\{0,1\}^n$, find a perceptron $(\mathbf{w}, w_0) \in \mathbb{R}^{n+1}$ that is consistent with T and has as few nonzero weights w_j, $1 \leq j \leq n$, as possible.

Obviously OPTIMUM PERCEPTRON is a special case of MIN RVLS with mixed inequalities.

While Lin and Vitter proved that OPTIMUM PERCEPTRON with binary inputs is NP-hard [115], Van Horn and Martinez established that the symmetric variant with strict inequalities is at least as hard to approximate as MIN DOMINATING SET [160]. Furthermore, they showed that an approximate algorithm which also minimizes the number of nonzero weights within a factor of $O(\log p)$ would require far fewer examples to achieve a given level of accuracy than any algorithm that does not minimize the perceptron size. More precisely, for such an Occam algorithm the number of training examples needed to PAC-learn would be almost linear in the minimum number of nonzero weights s. If $s = O(n^\alpha)$ for some $0 < \alpha < 1$, this is much less than the $O(n)$ examples required by a simplistic training procedure without size minimization.

By extending the nonapproximability bound for MIN RVLS$^=$ derived in theorem 3.3.9, we provide strong evidence that no such approximate algorithm exists.

Theorem 4.4.2 OPTIMUM PERCEPTRON *cannot be approximated within any constant factor unless* P = NP, *and within a factor of* $2^{\log^{1-\varepsilon} s}$ *for any* $\varepsilon > 0$ *unless* NP $\not\subseteq$ DTIME$(s^{\text{polylog}\, s})$, *where* $s = \max\{n, p\}$ *is the maximum of the number of inputs and the number of examples in the task.*

PROOF We adapt the reduction from SET COVERING with exact covers used in the proof of theorem 3.3.9 to the case of perceptrons with real weights.

To show that OPTIMUM PERCEPTRON cannot be approximated within any constant factor unless P = NP, we proceed like in the proof of proposition 3.3.6.

For any particular instance $(S, \mathcal{C})$ of SET COVERING, we construct $|S|$ positive examples with n components corresponding to the system

$$A\mathbf{w} > w_0 \, \mathbf{1}, \tag{4.18}$$

where $a_{ij} = 1$ if the ith element of S belongs to C_j and 0 otherwise. Furthermore, we include the negative example $\mathbf{0}$ ensuring that $w_0 \geq 0$. Hence we have a task with $|S|$ positive and one negative examples.

Clearly, the nonzero components of any weight vector $\mathbf{w} \in \mathbb{R}^n$ that correctly classifies all examples define a cover. Conversely, given any cover $\mathcal{C}'$ of cardinality K, the weight vector $\mathbf{w}$ given by

$$w_j = \begin{cases} 1 & \text{if } C_j \in \mathcal{C}' \\ 0 & \text{otherwise} \end{cases}$$

together with the threshold $w_0 = 0$ correctly classifies all examples and has K nonzero components. Therefore the minimum number of nonzero weights is either K or at least $\lfloor c \cdot K \rfloor$ depending on whether the corresponding SAT instance is satisfiable or unsatisfiable.

The constant gap between the satisfiable and unsatisfiable cases can then be increased recursively by self-improvement like in the proof of theorem 3.3.9. Since in the reduction the number of examples p is polynomially related to the number of components n, the same nonapproximability bound is valid with respect to both quantities. $\square$

Thus designing close-to-minimum size networks in terms of nonzero weights is very hard even for linearly separable tasks that are performable by single-layer networks. Clearly, the general problem for multilayer networks is at least as hard as OPTIMUM PERCEPTRON. Note that, since our bound has been established for single perceptrons, the design problem does not become easier even if we assume that the number of units contained in each layer of an optimum network is given and we only need to find an appropriate set of weight values. From a practical point of view, this implies that one should look at best for efficient heuristics with good average behavior.

Finally, it is worth recalling that, as mentioned in section 1.2.5, our discussion on OPTIMUM PERCEPTRON also pertains to the variable selection and parsimony problems in discriminant analysis [109].

Chapter 5

Training single perceptrons

In this chapter we study the fundamental problem of training single perceptrons. The focus is on several variants of the famous perceptron algorithm for maximizing consistency or, equivalently, minimizing disagreement. As mentioned in section 1.2.6, such methods are of considerable interest also for multilayer networks. Indeed they are extensively used in the methods of learning by choice of internal representations [79, 78] and in the important class of constructive algorithms that will be discussed in chapter 6. Since the global performances of a network depend on how well each unit performs the subtask it has been assigned, efficient algorithms for training perceptrons are needed. According to the complexity results in chapter 4, the best we can do is to devise efficient heuristics. In this framework, we are interested in simple methods providing reasonably good solutions in a short time.

After a brief overview of the perceptron algorithm, the thermal variant is revisited in section 5.2. Section 5.3 provides an interpretation in terms of error function minimization. Section 5.4 describes some simple extensions that lead to significant improvements. In section 5.5 a new probabilistic variant of the perceptron algorithm is proposed to deal with nonlinearly separable tasks. Unlike the thermal perceptron, this procedure is guaranteed to find an optimal weight vector with probability approaching 1 when the number of iterations tends to infinity. In section 5.6 a set of experimental results are reported and discussed. Finally, section 5.7 contains various general remarks.

5.1 The perceptron algorithm

As previously seen, training a perceptron amounts, given a task $T = \{(\mathbf{a}^k, b^k)\}_{1 \le k \le p} \subseteq \mathbb{R}^n \times \mathbb{B}$, to finding a vector $\mathbf{w}$ that satisfies as well as possible the corresponding linear system

$$\begin{aligned} \mathbf{a}^k\,\mathbf{w} > 0 \ \ \forall k \text{ such that } \ b^k = 1 \\ \mathbf{a}^k\,\mathbf{w} \le 0 \ \ \forall k \text{ such that } \ b^k = -1. \end{aligned} \tag{5.1}$$

The well-known *perceptron algorithm* (PA) proposed by Rosenblatt [143], starts with an arbitrary initial vector $\mathbf{w}_0$, randomly selects a sequence of input vectors $\mathbf{a}^k$ and at each iteration updates the current vector $\mathbf{w}_i$ as follows:

$$\mathbf{w}_{i+1} := \begin{cases} \mathbf{w}_i + \eta_i\ b^k\mathbf{a}^k & \text{if } \mathbf{a}^k \text{ is misclassified by } \mathbf{w}_i \\ \mathbf{w}_i & \text{otherwise,} \end{cases} \tag{5.2}$$

where the gain parameter η_i evolves through the iterations. This simple procedure is in fact a version of the Agmon-Motzkin-Schoenberg relaxation method for solving systems of linear inequalities where all the inequalities but one are relaxed at each iteration [3, 133].

It is worth noting that this procedure can be viewed as a gradient-like method for minimizing the error function

$$J_p(\mathbf{w}) = -\sum_{k \in \mathcal{M}(\mathbf{w})} b^k\mathbf{a}^k\mathbf{w}, \tag{5.3}$$

where $\mathcal{M}(\mathbf{w})$ denotes the set of indices of the input vectors that are misclassified by $\mathbf{w}$ [54]. Since the deviations $b^k\mathbf{a}^k\mathbf{w} \le 0$ for all $k \in \mathcal{M}(\mathbf{w})$, $J_p(\mathbf{w})$ is nonnegative and equal to zero for any $\mathbf{w}$ which performs the task correctly. In fact, the perceptron algorithm minimizes $J_p(\mathbf{w})$ using a stochastic gradient technique: the input vectors $\mathbf{a}^k$ are randomly chosen and the current weight vector $\mathbf{w}_i$ is modified after each presentation of an $\mathbf{a}^k$.

If the task is linearly separable and the gain parameter sequence $\{\eta_i\}_{i \ge 1}$ satisfies the following conditions

$$\eta_i \ge 0, \qquad \lim_{l \to \infty} \sum_{i=1}^{l} \eta_i = \infty \quad \text{ et } \quad \lim_{l \to \infty} \frac{\sum_{i=1}^{l} \eta_i^2}{\left(\sum_{i=1}^{l} \eta_i\right)^2} = 0, \tag{5.4}$$

the perceptron algorithm is guaranteed to yield a weight vector $\mathbf{w}$ that correctly classifies all $\mathbf{a}^k$ after a finite number of iterations. However, in the worst-case it requires an exponential number of iterations [130].

Whenever the task is nonlinearly separable, we look for an *optimal* weight vector that maximizes consistency. In this very frequent case, the above error-correction procedure does not terminate because some input vectors are always incorrectly classified. The algorithm can even go in a single iteration from an optimal weight vector to a worst-possible one. Nevertheless, this procedure has been proved to be bounded, i.e. the norm of the visited weight vectors is bounded by a constant that does not depend on the initial vector $\mathbf{w}_0$ [39]. This result is often referred to as the *cycling* theorem.

In [65] Gallant proposed a very simple extension of the PA, named the *pocket* algorithm, that makes it well behaved for nonlinearly separable tasks. The idea is just to run the perceptron and memorize the best visited solution. If $\mathbf{w}_M$ denotes the best weight vector found so far, the current $\mathbf{w}_i$ can be tested in several ways. The simplest but extremely time-consuming approach is to check for every $\mathbf{w}_i$ whether it makes fewer errors than $\mathbf{w}_M$. A more convenient technique is to compute the total number of errors only when the current $\mathbf{w}_i$ correctly classifies more consecutive input vectors than $\mathbf{w}_M$. If l_M denotes the number of consecutive correct classifications for $\mathbf{w}_M$ and l_i the corresponding number for $\mathbf{w}_i$, the total number of errors made by $\mathbf{w}_i$ is computed and compared to that of $\mathbf{w}_M$ only if $l_i > l_M$. The interesting property of these variants is that they have an asymptotic convergence guarantee for any task with integer inputs. More precisely, the probability that $\mathbf{w}_M$ is optimal tends to 1 when the number of iterations tends to infinity [65]. Furthermore, since only integer tasks are considered, the advantage of integer arithmetic can be exploited when an integer gain parameter η_i is used.

It is worth recalling that an enumeration technique like Greer's algorithm is guaranteed to find an optimal solution in a finite but exponential amount of time. However, this type of methods are not suited to adaptive learning systems because they involve complex computations and they are only applicable to small-sized problems. Moreover, they are not easily adaptable to on-line learning in which the whole task is not known in advance and some examples become available during the training phase.

5.2 The thermal perceptron algorithm revisited

For maximizing perceptron consistency, Frean proposed another variant of the PA named *thermal perceptron algorithm* (TPA) [63]. Although the TPA has no guarantee of convergence, it performs better experimentally than the pocket method for linearly as well as nonlinearly separable tasks. If its "temperature parameter" is reasonably well tuned, it also compares favorably with other alternative methods based on least mean squared [165] or cross-entropy minimization [63, 86].

The basic idea is to favor weight updates that aim at correcting errors whose *total input* $v^k = b^k \mathbf{a}^k \mathbf{w}$ is relatively close to zero. Indeed, since v^k is proportional to the distance from the input to the hyperplane $H = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{wx} = 0\}$ defined by the current $\mathbf{w}$, the large weight changes that would be required to correct errors with a large $|v^k|$ are likely to corrupt other well classified $\mathbf{a}^k$. Therefore the magnitude of the weight modifications decreases exponentially with $|v^k|$ and depends on a "temperature" parameter t. Starting with an arbitrary initial nonzero $\mathbf{w}_0$, the weight vector is then updated as follows:

$$\mathbf{w}_{i+1} := \begin{cases} \mathbf{w}_i + \frac{t}{t_0} \exp\left(\frac{b^k \mathbf{a}^k\ \mathbf{w}_i}{t}\right) \cdot b^k \mathbf{a}^k & \text{if } \mathbf{a}^k \text{ is misclassified by } \mathbf{w}_i \\ \mathbf{w}_i & \text{otherwise,} \end{cases} \tag{5.5}$$

where t linearly decreases from an initial value t_0 to 0 in a given number of cycles $c_{\max}$ through the task. This is achieved by setting at the beginning of the cth cycle $t := \gamma t_0$ with $\gamma = 1 - c/c_{\max}$. Note that the annealing process allows to stabilize the weight vectors over any given training period. For any fixed temperature t, the perceptron convergence theorem can immediately be extended to the case with exponential updates [61]. However, no convergence proof is known for the case with annealing scheme.

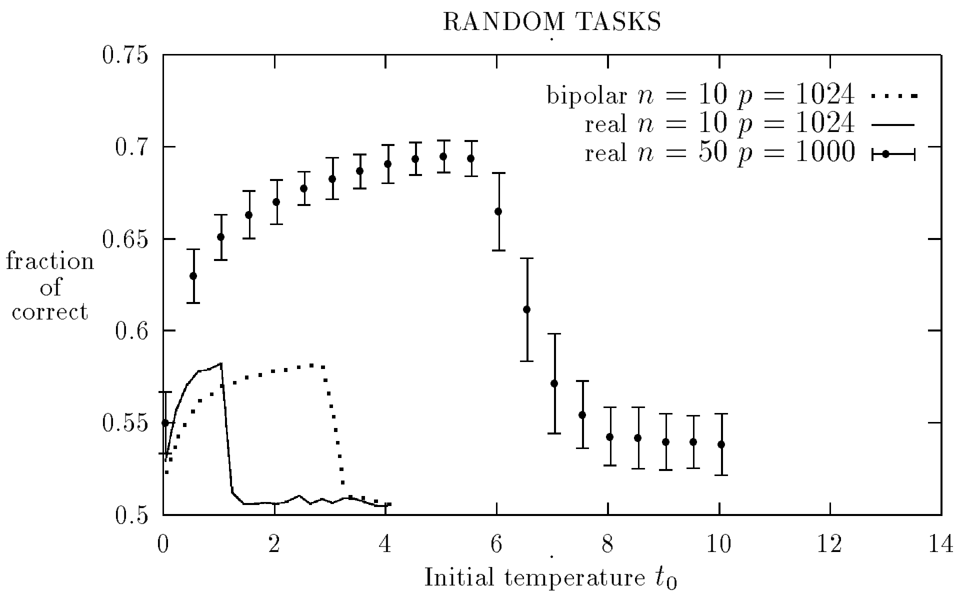


Figure 5.1: Random tasks with different sizes and different types of inputs. The percentage of correctly classified input vectors for various values of the initial temperature t_0. Typical standard deviations are indicated by vertical bars.

As briefly pointed out in [63], choosing a good value for the initial temperature t_0 is a major challenge because it has a strong impact on the quality of the solutions. Typical situations for highly nonlinear tasks with real or bipolar inputs are illustrated in Figure 5.1. These tasks are generated by randomly selecting the input vectors in (or on) the unit hypercube and by assigning them an output 1 or -1 with equal probability. The fraction of correct classifications after 150 cycles is plotted against the initial temperature t_0. These are averages over 100 random tasks. For tasks with $n = 50$ real inputs and $p = 1000$ examples, an initial temperature between 3.0 and 6.0 is adequate. For the same type of tasks but different sizes $n = 10$ and $p = 1024$, values between 0.45 and 1.5 are recommended. If the same sizes are considered, but the inputs are restricted to take bipolar values, the satisfactory range for t_0 is still different. Since t_0 depends on the type as well as on the sizes of T, an appropriate value must be determined for each task. Although an explicit search process is too time-consuming, a systematic way of selecting the initial temperature is needed in order to exploit the real potentialities of the thermal approach. Indeed, reasonably good values of t_0 are required to outperform the methods that minimize the least-mean-squared or cross-entropy error [63].

Intuitively t_0 should be of the same order of magnitude as the typical values of the total inputs v^k. Since the range of the v^k can vary enormously, the range of the temperature must be adapted accordingly during the training process. A similar suggestion was mentioned by Burgess et al. in [42], where they estimate the typical value of v^k to be $\langle |v^k| \rangle = \sum_{k=1}^{p} |v^k|/p$. In order to try to attenuate oscillations and to keep t_0 close to $\langle |v^k| \rangle$, they start with $t_0 = 2\langle |v^k| \rangle$ and set $t_0 := \frac{2}{3}(t_0 + \langle |v^k| \rangle)$ after each *cycle* through the task. The fluctuations of $\langle |v^k| \rangle$ are taken into account by setting $t := \gamma t_0$, where γ decreases linearly from 1 to 0 during the $c_{\max}$ cycles. Note that updating t_0 amounts to rescaling t with respect to the current $\langle |v^k| \rangle$ without affecting the linear decrease induced by γ.

Unfortunately, this temperature adaptation scheme has a negative effect on the overall behavior of the iterative procedure. In Figure 5.2 the percentage of errors is plotted versus the number of cycles for five different values of $c_{\max}$. The characteristic shape of these curves clearly shows that the procedure is not effective: the fraction of errors remains approximately constant during more than 90% of the cycles. In terms of annealing processes, this indicates a too long "heating" phase during which convergence cannot take place. The same type of behavior has been observed for all nonlinearly separable tasks we have considered. This is shown in the first two figures of section 5.6. The dotted curve of Figure 5.2 that decreases more rapidly corresponds to Frean original version with a near-optimal initial temperature $t_0 = 0.8$. It is worth noting that, if the initial temperature

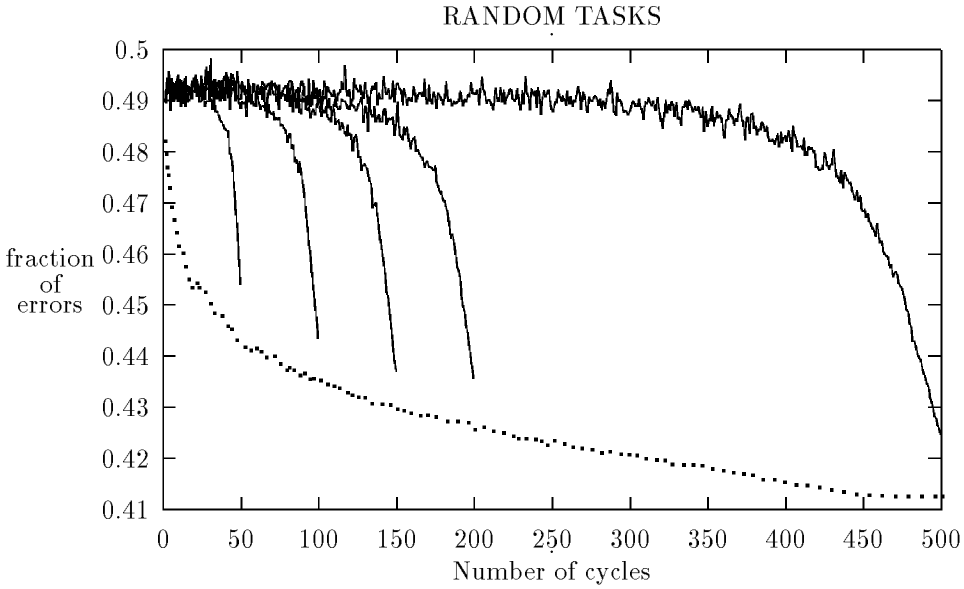


Figure 5.2: The five upper lines are typical convergence curves of the thermal perceptron algorithm using Burgess et al. temperature adaptation scheme for various choices of $c_{\max}$. Averages are over 100 random tasks with $n = 10$ real inputs and $p = 1000$ examples. The dotted line corresponds to Frean TPA version with a near-optimal initial temperature $t_0 = 0.8$.

is taken to be $t_0 = 2\langle|v^k|\rangle$, Frean version performs very poorly. In this case, the average fraction of errors remains higher than 0.47 during 500 cycles.

According to the above observations, decreasing γ linearly turns out to be inappropriate when t_0 is updated as a linear combination of t_0 and $\langle|v^k|\rangle$.

5.3 The TPA error function

Although in [61, 63] the thermal perceptron algorithm is contrasted with the methods that optimize a criterion using a gradient descent technique, the TPA can be viewed as a procedure for minimizing an error function which varies during the annealing process.

For any fixed value of the temperature t, we consider the error function

$$J_{tp}(\mathbf{w}) = \sum_{k \in \mathcal{M}(\mathbf{w})} \exp\left(\frac{b^k \mathbf{a}^k \mathbf{w}}{t}\right) \tag{5.6}$$

where $\mathcal{M}(\mathbf{w})$ denotes, as in section 5.1, the set of indices of the misclassified input vectors. This function is similar to the perceptron criterion (5.3).

Since the gradient of J_{tp} with respect to $\mathbf{w}$ is given by

$$\nabla J_{tp}(\mathbf{w}) = -\sum_{k \in \mathcal{M}(\mathbf{w})} \exp\left(\frac{b^k \mathbf{a}^k \mathbf{w}}{t}\right) \cdot \frac{b^k \mathbf{a}^k}{t},$$

the thermal perceptron procedure can be interpreted in terms of J_{tp} minimization. More precisely, a TPA update amounts to an iteration of stochastic gradient descent aiming at minimizing (5.6) and is given by

$$\Delta \mathbf{w} = \begin{cases} \eta_i \exp\left(\frac{b^k \mathbf{a}^k \mathbf{w}_i}{t}\right) \cdot \frac{b^k \mathbf{a}^k}{t} & \text{if } \mathbf{a}^k \text{ is misclassified by } \mathbf{w}_i \\ 0 & \text{otherwise,} \end{cases} \tag{5.7}$$

where the gain parameter is $\eta_i = \frac{t^2}{t_0}$.

A posteriori, the choice of the criterion is quite natural: $J_{tp}(\mathbf{w})$ is nonnegative and equal to 0 only if $\mathbf{w}$ performs the task correctly. Due to the exponential, the misclassified $\mathbf{a}^k$ with a small $|v^k|$ are strongly penalized with respect to those with a large $|v^k|$. Since $\mathbf{a}^k\mathbf{w}$ is proportional to the distance from $\mathbf{a}^k$ to the current hyperplane H defined by the $\mathbf{w}$, this favors corrections for input vectors that are close to H.

Unlike J_p, J_{tp} does not only depend on the weight vector $\mathbf{w}$ but also on the temperature t. It is interesting to point out that when t tends to infinity $J_{tp}(t)$ converges pointwise towards the function which counts the number of errors. This criterion, denoted by $E(\mathbf{w})$, is the function we really want to minimize but it cannot be used with a gradient-like technique because it is piecewise constant. The standard perceptron algorithm is clearly recovered at high t. Conversely, when the temperature t tends to 0 the function $J_{tp}(t)$ converges towards the constant function 0.

Since the temperature is decreased after each cycle, the error function associated to the initial t_0 evolves step by step towards the constant function 0 and the thermal perceptron algorithm minimizes over this finite sequence of functions. Starting with an error function $J_{tp}(t_0)$, which can be viewed as a coarse approximation of the true objective function $E(\mathbf{w})$, p stochastic gradient descent steps are performed on the considered $J_{tp}(t)$. Clearly, any annealing scheme where t decreases to 0 produces stable weight vectors because the final error function associated to $t = 0$ has a zero gradient.

This interpretation partially accounts for the fact that the best results are obtained for intermediate values of the initial temperature. While the error functions corresponding to large initial t_0 are close to the badly-behaved $E(\mathbf{w})$ criterion, those associated to small values are close to the constant zero function which does not provide any useful information. Not surprisingly, the way the temperature t decreases, and hence the error function varies, has a strong impact on the quality of the resulting weight vector.

5.4 Two adaptive and efficient TPA versions

According to the observations in section 5.2, we decrease the multiplicative factor γ more rapidly than linearly and we modify accordingly the way t_0 is updated. At the beginning of the cth cycle, we set $\gamma := 1/c$, $t_0 := \frac{1}{3}t_0 + \frac{2}{3}\langle|v^k|\rangle$, $t := \gamma t_0$ and $\eta_i := t/t_0 (= \gamma)$. Thus the average of the v^k has more importance and the two coefficients add to 1. The specific values were determined experimentally as being the most efficient among a number of possibilities [53]. As usually, the initial weight vector is generated at random with uniform components in the interval $[-1, 1]$.

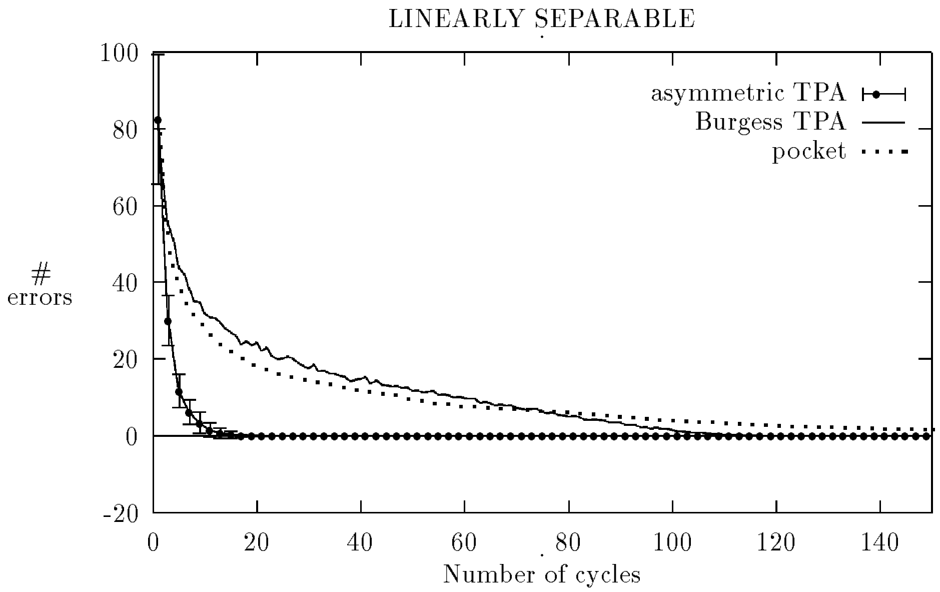


Figure 5.3: Average number of errors for linearly separable tasks with $n = 25$ real inputs and $p = 1000$ examples.

Figure 5.3 shows typical results for linearly separable tasks. These tasks are constructed by picking a random weight vector $\mathbf{w}$, generating random input vectors in the unit hypercube and assigning them the corresponding output. The procedure guarantees that the two classes have the same number of examples. All results are averages over 100 tasks with $n = 25$ inputs and $p = 1000$ examples. Our adaptive version of the TPA finds an optimal weight vector for all 100 tasks after only 21 cycles while the variant suggested by Burgess et al. needs about 125 cycles. The pocket curve corresponds to a linear decrease of the gain parameter. If $\eta_i = 1/c$, the pocket procedure converges as fast as the asymmetric

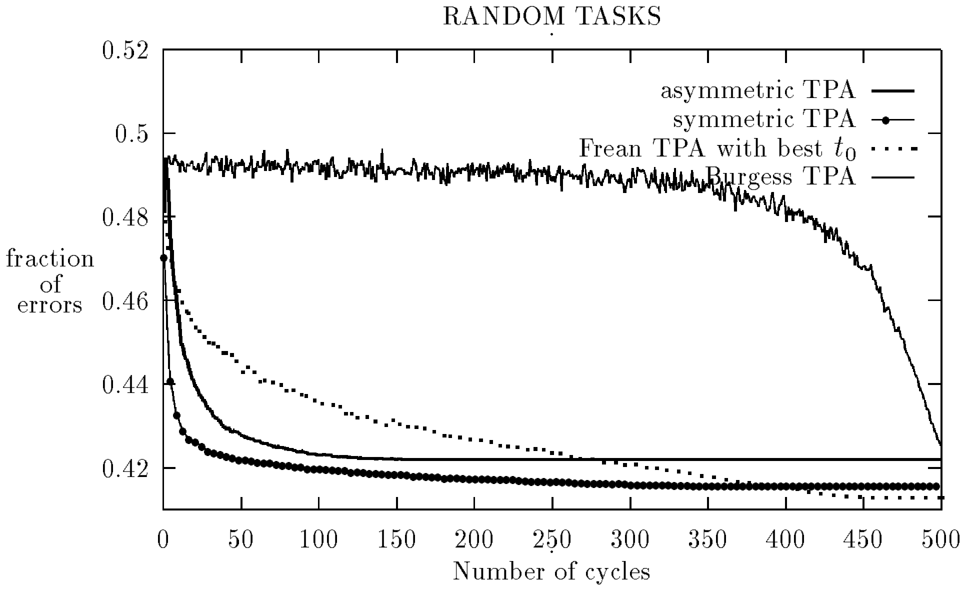


Figure 5.4: Percentage of errors for random tasks with $n = 10$ bipolar inputs and $p = 1024$ examples. Averages over 100 tasks.

TPA during the first 10 cycles but then it saturates at an average number of errors of 1.73. As for the asymmetric TPA, all standard deviations are small.

Within an annealing approach, one may ask whether it is worthwhile to reconsider also the input vectors that are correctly classified and lie close to the corresponding hyperplane. The rationale is that they may help convergence by providing useful information on the distribution of the two classes. This leads to the *symmetric* variant of the TPA where the current weight vector $\mathbf{w}_i$ is updated as follows:

$$\mathbf{w}_{i+1} := \mathbf{w}_i + \frac{t}{t_0} \exp\left(\frac{|b^k \mathbf{a}^k \ \mathbf{w}_i|}{t}\right) \cdot b^k \mathbf{a}^k \tag{5.8}$$

for any presented $\mathbf{a}^k$. Unlike the asymmetric version, the update does not depend on the sign of v^k. Since every $\mathbf{a}^k$ gives rise to a weight modification, this approach is not applicable to the standard perceptron algorithm because it would prevent convergence. However, it is suited to a thermal framework where only the input vectors lying sufficiently close to the current hyperplane H are considered. During the first cycles, all input vectors lead to a modification but, as the temperature decreases, the size of the "updating zone" is rapidly reduced.

In the symmetric TPA, we set at the beginning of the cth cycle $\gamma := 1/c$, $t_0 := \frac{1}{3}t_0 + \frac{2}{3}\langle|v^k|\rangle$ and $t := \gamma t_0$. But, instead of taking $\eta_i = t/t_0$ which would also decrease as $1/c$, we consider the linear scheme $\eta_i = 1 - c/c_{\max}$. This amounts to multiplying the exponential factor by a gain parameter that decreases less rapidly. In fact, the additional noise introduced by the symmetric updates favors a slower scheme for η_i.

As shown in Figure 5.4, the symmetric TPA converges faster than the asymmetric one for highly nonseparable tasks. It is worth noting that the weight vectors provided by our TPA variants are slightly better than those obtained with Burgess version and are comparable to those given by Frean version with the best initial temperature. Thus, our simple modifications make the TPA adaptive as well as efficient. Comparisons for other types of nonlinearly separable tasks are reported and discussed in section 5.6.

5.5 The probabilistic perceptron algorithm

In the thermal perceptron algorithm, the current weight vector is updated every time the presented input vector $\mathbf{a}^k$ is misclassified but the magnitude of the modification decreases exponentially in time. Thus, whatever the decreasing schedule for the parameter t is, the modifications become very small after a relatively small number of cycles and the updating process requires computations of higher and higher precision.

In order to avoid this pitfall while guaranteeing convergence, we propose a probabilistic approach. Instead of updating, by a small and rapidly decreasing amount, the current weight vector $\mathbf{w}_i$ whenever an error occurs, $\mathbf{w}_i$ is updated according to the amount prescribed by the standard PA but with a rapidly decreasing probability. In the asymmetric version of the *probabilistic perceptron algorithm* (PPA), the current weight vector $\mathbf{w}_i$ is modified with probability:

$$\Pr\{\mathbf{w}_{i+1} := \mathbf{w}_i + \eta_i b^k \mathbf{a}^k\} \quad = \quad \exp\left(\frac{b^k \mathbf{a}^k \mathbf{w}_i}{t}\right) \tag{5.9}$$

when the presented $\mathbf{a}^k$ is misclassified and remains unchanged when the presented $\mathbf{a}^k$ is correctly classified.

Each TPA variant is characterized by the product of the gain parameter η and the typical exponential factor e, which depends on the current temperature t. For all misclassified input vectors $\mathbf{a}^k$, the magnitude of the weight modification is proportional to $\eta_i e$. In order to avoid the need for very high precision computations, the probabilistic idea consists in updating only a small fraction e of the misclassified $\mathbf{a}^k$ by the usual amount $\eta_i b^k \mathbf{a}^k$ of the standard perceptron rule. In this case, the exponential factor is only used once to decide whether the update must be carried out or whether the current weight vector remains unchanged. Nevertheless, an extra pseudorandom number is needed whenever a misclassified input vector is considered.

In the symmetric version of the PPA, the current weight vector $\mathbf{w}_i$ is updated with probability:

$$\Pr\{\mathbf{w}_{i+1} := \mathbf{w}_i + \eta_i b^k \mathbf{a}^k\} \quad = \quad \exp\left(\frac{-|b^k \mathbf{a}^k \mathbf{w}_i|}{t}\right) \tag{5.10}$$

for any presented $\mathbf{a}^k$. Thus we have a nonzero probability of modifying the current $\mathbf{w}_i$ for all $\mathbf{a}^k$ with a small $|v^k|$ compared to t. Independently of the type of input vector, the probability of update decreases exponentially with $|v^k|$.

Unlike the TPA, both probabilistic variants have an asymptotic convergence guarantee if the best weight vector encountered is stored during the procedure. As in the *pocket* algorithm [65], the storage test can be performed either whenever the current weight vector $\mathbf{w}_i$ is updated or only when the number l_i of consecutive input vectors which have been correctly classified is such that $l_i > l_M$.

Before proving the convergence guarantee for the symmetric and asymmetric versions of the PPA, we give a simpler and stronger convergence proof for the pocket algorithm than the original one [65]. Our proof is simpler because it does not rely on the cycling theorem [39, 130] and more general because it holds for tasks whose input vectors may have real components, instead of being restricted to integer values. Recall that in [65] the cycling theorem and the integrality assumption were used to make sure that the pocket procedure visits only a bounded and therefore finite set of weight vectors. Moreover, the gain parameter was restricted to be constant $\eta_i = 1$.

Theorem 5.5.1 *Given an arbitrary task $T = \{(\mathbf{a}^k, b^k)\}_{1 \leq k \leq p} \subseteq \mathbb{R}^n \times \mathbb{B}$ and any probability $0 < \pi < 1$, there exists a positive integer i_π such that after i_π iterations of the pocket algorithm (with η_i satisfying the usual perceptron conditions) the probability that the stored weight vector $\mathbf{w}_M$ is optimal exceeds π.*

Proof We first verify the following basic but strong property: *There is a nonzero probability that starting from any weight vector the perceptron algorithm (or the pocket one) reaches an optimal weight vector after a finite number of iterations.* Let us consider an arbitrary optimal weight vector $\mathbf{w}^*$ together with the set of associations that it performs correctly, denoted by T^*. According to the perceptron convergence theorem [130], if only associations in the linearly separable subtask T^* are selected, we are guaranteed to reach, from any initial $\mathbf{w}$ and after a finite number of iterations, a weight vector that performs

T^* correctly. Such a weight vector, which is obviously optimal for the original task T, will be obtained after at most $B = \lceil \frac{\beta \|\mathbf{w}^*\|^2}{\gamma^2} \rceil$ updates, where $\beta = \max_{1 \leq k \leq p^*} \| b^k \mathbf{a}^k \|^2$ and $\gamma = \min_{1 \leq k \leq p^*} (b^k \mathbf{a}^k) \mathbf{w}^*$ are defined with respect to the subtask T^* containing p^* examples. Note that B does not depend on the weight vector we start with. Since the $\mathbf{a}^k$ are randomly selected with positive (but possibly different) probabilities and since there exists a finite number of possible input vector sequences of length B, an appropriate sequence with only vectors in T^* occurs with nonzero probability. Thus the probability that the perceptron algorithm leads from any given weight vector to an optimal one in at most B iterations is strictly greater than 0. This probability depends on the weight vector and different weight vectors may have different probabilities. In particular, optimal weight vectors have probabilities 1.

We now prove that starting with any weight vector *the number of times the perceptron algorithm (or the pocket one) visits an optimal weight vector converges to infinity with probability* 1 *as the number of iterations grows.* Therefore we need Kolmogorov's generalized version of the strong law of large numbers which holds for sequences of nonidentically distributed random variables.

Theorem 5.5.2 [153] *Let $X_1, X_2, \ldots$ be a sequence of independent random variables with means $\mu_1, \mu_2, \ldots$ and variances $\sigma_1^2, \sigma_2^2, \ldots$ If $\sum_{i=1}^{\infty} \sigma_i^2 / i^2$ converges, then with probability* 1

$$\frac{1}{l} \sum_{i=1}^{l} X_i - \frac{1}{l} \sum_{i=1}^{l} \mu_i \rightarrow 0 \tag{5.11}$$

when $l \rightarrow \infty$.

This result is applied to the sequence of Bernoulli variables associated to the weight vectors $\{\mathbf{w}_i\}_{i \geq 1}$ generated during a run of the pocket procedure, which indicate whether they yield an optimal weight vector in finite time. More precisely, for any $i \geq 1$ the variable X_i takes the value 1 if the pocket algorithm leads from $\mathbf{w}_{i-1}$ to an optimal weight vector in at most B iterations and 0 otherwise. All assumptions of the above theorem are easily verified. The X_i are independent, their means μ_i are equal to their respective probabilities of success π_i and their variances σ_i^2 amounts respectively to $\pi_i(1 - \pi_i)$. Since all variances are bounded above by $1/4$, $\sum_{i=1}^{\infty} \sigma_i^2 / i^2$ converges.

According to the definition of *almost sure convergence*, the strong law of large numbers states that

$$\Pr\{\lim_{l \rightarrow \infty} (\frac{1}{l} \sum_{i=1}^{l} X_i - \frac{1}{l} \sum_{i=1}^{l} \pi_i) = 0\} = 1. \tag{5.12}$$

Thus, since

$$\lim_{l\to\infty} \inf(\frac{1}{l}\sum_{i=1}^{l}\pi_i) > 0, \tag{5.13}$$

we have

$$\lim_{l\to\infty}(\frac{1}{l}\sum_{i=1}^{l}X_i) > 0 \tag{5.14}$$

with probability 1, which implies that the number of times an optimal weight vector is visited during the perceptron procedure converges to infinity with probability 1[1]. Indeed, if the sequence $\sum_{i=1}^{l} X_i$ were bounded then $\frac{1}{l}\sum_{i=1}^{l} X_i$ would tend to 0 when $l \to \infty$.

So far we have proven that, as the number of iterations grows, the perceptron algorithm (or the pocket one) will almost surely visit an optimal weight vector for a nonzero fraction of the time. However, the visited optimal weight vectors are not guaranteed to be detected and stored because the total number of errors is computed only if the longest run of consecutive input vectors that are correctly classified is longer than l_M.

To complete the proof, we need to observe that

$$\Pr\{l_i \geq l_M \mid \mathbf{w}_i \in W^*\} > 0 \tag{5.15}$$

for any given $\mathbf{w}_M$, where W^* denotes the set of all optimal weight vectors. In fact, there is a nonzero probability that the input vectors of T which are correctly classified by an optimal $\mathbf{w}_i$ are presented consecutively and hence that l_i is equal to the largest number of associations in T which can be performed simultaneously. Thus any visited optimal weight vector has a nonzero probability to be detected and to be stored if the current $\mathbf{w}_M$ is suboptimal. Since the number of times that optimal weight vectors are visited diverges with probability 1, the probability that none of them is detected decreases to 0 when the number of iterations grows. In other words, the probability that $\mathbf{w}_M$ is optimal converges to 1 when the number of iterations tends to infinity. □

An interesting aspect of this convergence proof for the pocket algorithm is that it can be easily extended to both cases with symmetric and asymmetric updates. This is not true for the original proof because the cycling theorem does not hold for the symmetric variant.

Corollary 5.5.3 *Given an arbitrary task $T = \{(\mathbf{a}^k, b^k)\}_{1\leq k\leq p} \subseteq \mathbb{R}^n \times \mathbb{B}$ and any probability $0 < \pi < 1$, there exists a positive integer i_π such that after i_π iterations of the probabilistic*

[1]This is equivalent to $\Pr\{\lim_{l\to\infty}\sum_{i=1}^{l} X_i = \infty\} = 1$.

perceptron algorithm (with η_i satisfying the usual conditions and any temperature scheme with $t > 0$) the probability that the stored weight vector $\mathbf{w}_M$ is optimal exceeds π.

PROOF When asymmetric probabilistic updates are performed and the temperature parameter t takes a fixed value strictly greater than 0, it suffices to note that the basic property of the above proof remains valid. Indeed, there is a nonzero probability that from any weight vector the procedure reaches an optimal one after a finite number of iterations. These probabilities are smaller than those for the pocket algorithm because the corrections are performed according to the exponential probabilities but they remain strictly greater than 0. Thus the strong law of large numbers is still applicable.

It is worth noting that this result holds also for arbitrary reduction schemes of the temperature t. As a matter of fact, t may decrease but it must converge to a value that is strictly greater than zero.

The same guarantee can readily be extended to the case of symmetric probabilistic updates. The probability of visiting an optimal weight vector from any initial weight vector is still nonzero and the other arguments apply. □

Thus, unlike the TPA, the probabilistic and the pocket perceptron algorithms have the same asymptotic and probabilistic convergence guarantees. However, as we will see in the next section, the PPA variants perform much better experimentally than the pocket algorithm since they behave very similarly to the corresponding TPA versions.

As in the thermal case, we have experimented with various decreasing schemes for γ and η_i and found that setting $t := \gamma t_0$ where $\gamma = 1/c$ and taking $\eta_i := 1 - c/c_{\max}$ is a good choice. The initial temperature t_0 is updated after each cycle according to $t_0 := \frac{1}{2}t_0 + \langle |v^k| \rangle$.

5.6 Comparative results

We first consider linearly separable tasks. As shown in Table 5.1, the asymmetric TPA and PPA find a zero error weight vector much faster than Burgess version. Note that Frean TPA with $t_0 = 2\langle |v^k| \rangle$ never achieves zero error on all 100 tasks. Nevertheless, the average number of errors left after 150 cycles is in general smaller than 0.1. In the linearly separable case, the symmetric variants do not work as well as the asymmetric ones. For all sizes of Table 5.1, the average number of errors over the 100 tasks was always sensibly larger than zero, but at most equal to 2.3 with a standard deviation smaller than 2.0.

n	p	Burgess TPA	asym TPA	asym PPA
25	500	110	21	59
	1000	125	22	23
	2000	134	26	28
	3000	141	39	37
50	500	71	32	84
	1000	101	21	22

Table 5.1: Linearly separable tasks. Number of cycles needed to find an optimal weight vector for 100 different tasks. Various sizes n and p are considered.

This might be due to the additional "noise" that is introduced when the weight vector is updated at each iteration while a correctly classified input vector is presented.

The situation is quite different for nonseparable tasks. Two typical kinds of nonlinearly separable training sets are considered. Unlike the previously mentioned random cases, both of them have a particular structure.

We first focus on nearly separable tasks that contain at most s percent of errors, where $0 < s \leq 10$. This corresponds to the common situation in which the examples are almost linearly separable but a given percentage s of *outliers* are present. Such tasks can be simply constructed from linearly separable ones with $\lfloor p(1 - s/100) \rfloor$ examples. It suffices to select $\lceil p\, s/100 \rceil$ new random input vectors and to assign them the opposite output with respect to the considered weight vector [63]. Notice that, since some outliers may also be correctly classified, the solution found can occasionally make slightly less than the prescribed fraction of errors. Figure 5.5 shows the convergence curves for tasks with at most 10% of outliers, $n = 25$ real inputs and $p = 1500$ examples. As for random training sets, our two TPA variants and Burgess version provide comparable solutions but the former ones find them much faster. Not surprisingly, the symmetric TPA that converges more rapidly saturates at a slightly higher number of errors than the asymmetric version.

Since the final number of errors is almost the same, the question arises as to whether Burgess version might give better solutions when a larger number of cycles is considered. Typical results for tasks with at most 5% of outliers are reported in Table 5.2. After 500 cycles, the solutions are comparable with a slight advantage for the asymmetric TPA. As usually, the latter method converges considerably faster. The same kind of results were observed in all our experiments over 500 cycles. At this point, it is worth recalling that

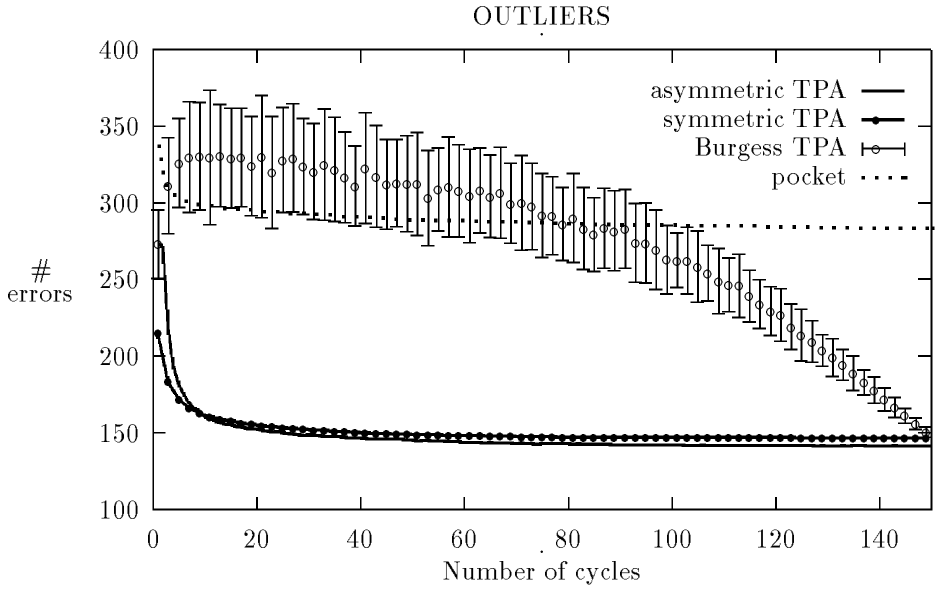


Figure 5.5: Averages over 100 tasks with at most 10% of outliers, $n = 25$ real inputs and $p = 1500$ examples.

p	Burgess TPA average	asym TPA iteration	asym TPA average
500	41.8 ± 2.68	77	40.9 ± 2.79
1000	92.7 ± 2.52	52	91.1 ± 2.74
1500	144.0 ± 2.26	60	141.5 ± 2.41
2000	194.7 ± 2.93	69	190.9 ± 2.66
2500	245.3 ± 3.27	76	240.9 ± 2.82

Table 5.2: Nearly separable tasks with at most 5% of outliers. Number of cycles needed to find (in average) a weight vector that makes the same number of errors as the average one obtained by Burgess TPA after 500 cycles. The mean number of errors after 500 cycles are mentioned in the second and third columns. Averages are over 100 different tasks.

we are mainly interested in finding reasonably good solutions in a relatively short amount of time.

Of course, if the Burgess TPA is applied for a much longer number of cycles, better solutions might be expected. The situation is slightly different for the asymmetric and the symmetric TPA. Since the control parameter γ decreases more rapidly, the magnitude of the weight modifications becomes smaller and a saturation phenomenon occurs after a certain number of cycles. In order to exploit the good convergence properties, one can use the same type of annealing schemes for γ and η_i over a given period (e.g. 100 cycles) and restart them periodically. By progressively decreasing the successive starting values for γ and η_i, the overall annealing process can be stabilized over a fixed number of periods.

The second type of nonlinearly separable tasks are generated from two overlapping Gaussian distributions with variances 1 and covariances 0. The centers are randomly picked within the unit hypercube and then rescaled so as to be at a distance 1.

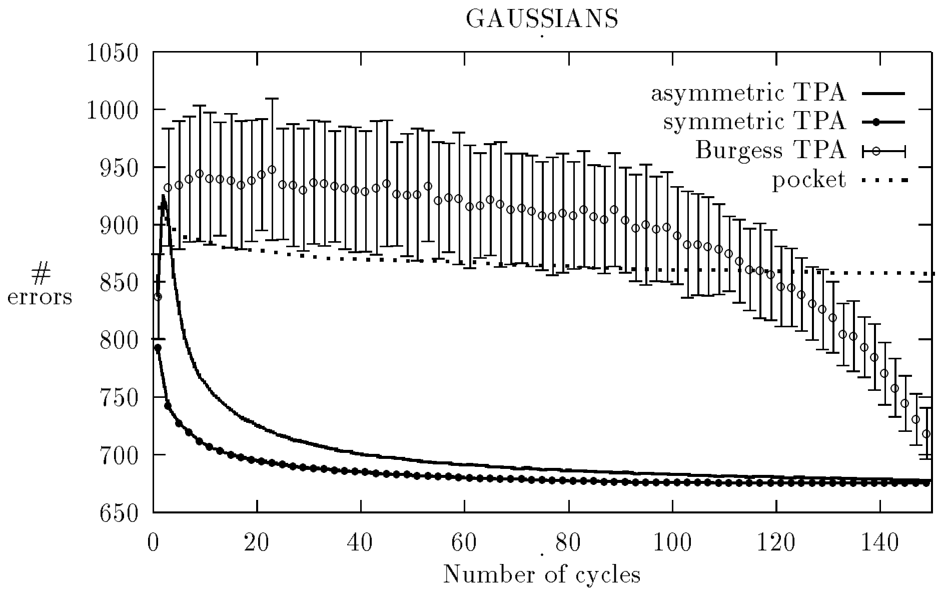


Figure 5.6: Overlapping Gaussian distributions. Averages over 100 tasks with $n = 30$ real inputs and $p = 2500$ examples.

Figure 5.6 illustrates the situation for Gaussian tasks with $n = 30$ inputs and $p = 2500$ examples. As in the previous cases, both our TPA variants converge considerably faster than Burgess TPA and give slightly better solutions. The peak occurring during the first cycles of the asymmetric TPA is probably related to the average value of v^k. At the beginning, $\langle |v^k| \rangle$ tends to be large and, since t_0 is a linear combination of the previous t_0 and $\langle |v^k| \rangle$, it drives t_0 to large values. However, this is rapidly compensated by the faster decrease of γ. Finally, notice that, as for all the other types of tasks, the pocket procedure performs very poorly.

The results obtained for Gaussian tasks of different sizes are reported in Table 5.3.

Although the symmetric and asymmetric probabilistic variants do not converge as fast as the corresponding TPA versions, they have interesting convergence properties compared to Burgess TPA. Thus, if the task is likely to be linearly separable and a fairly good solution

p	Burgess TPA average	asym TPA	asym PPA	sym TPA	sym PPA
1000	251.1 ± 12.5	28	76	19	110
1500	405.6 ± 16.0	27	57	11	33
2000	557.8 ± 17.5	29	63	10	26
2500	713.2 ± 21.8	28	56	8	20
3000	866.8 ± 24.7	26	55	7	17

Table 5.3: Overlapping Gaussian distributions. Number of cycles needed to find (in average) a weight vector that makes the same number of errors as the average one obtained by Burgess TPA after 150 cycles. The mean number of errors is also mentioned. Averages are over 100 different tasks.

is needed within a small number of cycles, the symmetric versions are recommended. Otherwise, the asymmetric ones should be used.

Figures 5.7 and 5.8 compare the probabilistic and thermal approaches. Recall that the probabilistic procedures aim at simulating small modifications of the weights with larger ones that occur with a small probability. These typical results show that the probabilistic variants and the corresponding TPA versions behave in a similar way. These curves correspond to identical annealing schemes, namely $\gamma := 1/c$, $t_0 := 0.5t_0 + \langle |v^k| \rangle$, $t := \gamma t$ and $\eta_i := t/t_0 (= \gamma)$. For different choices of these parameters, the difference between the probabilistic and thermal variants can be larger. According to Tables 5.1 and 5.3, the PPA versions exhibit fast convergence properties for linearly separable as well as Gaussian tasks.

It is worth noting that, unlike the pocket method, the probabilistic algorithms provide good solutions even if the "pocket" device is not used. In other words, one does not need to store the visited weight vector with the longest sequence of consecutive correctly classified $\mathbf{a}^k$. Since all PPA results reported in this chapter have been obtained without a "pocket", this indicates that such procedures tend to remain in good regions of the weight space.

Finally, we consider a well-known real-world problem that has been extensively used in

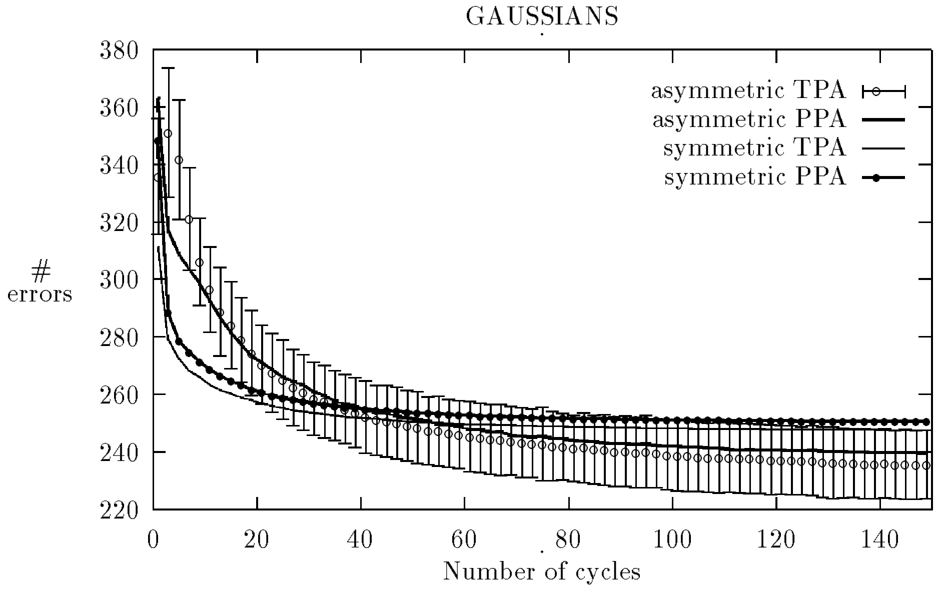


Figure 5.7: Comparison between symmetric and asymmetric TPA and the corresponding probabilistic variants. Gaussian tasks with $n = 30$ real inputs and $p = 1000$ examples.

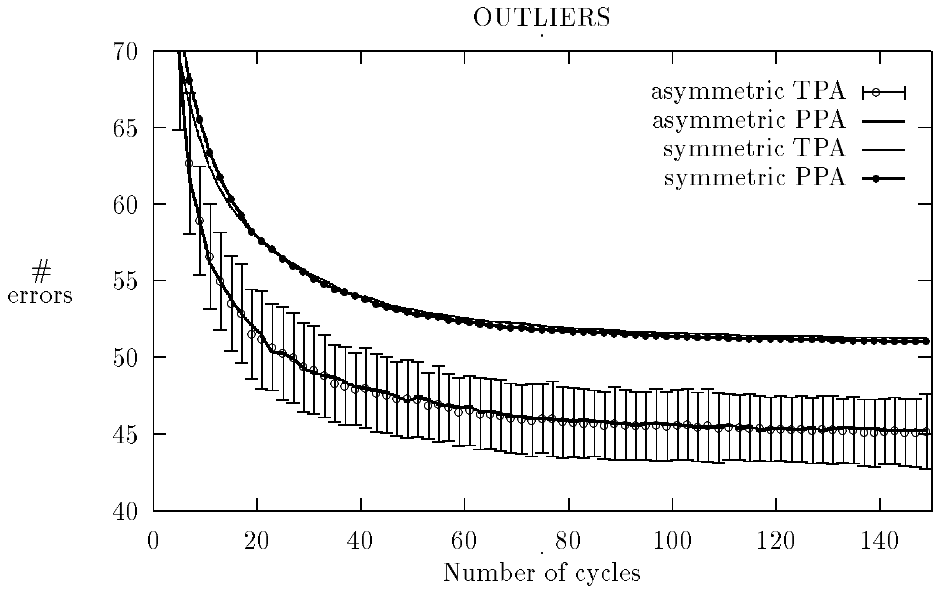


Figure 5.8: Comparison between symmetric and asymmetric TPA and the corresponding probabilistic variants. Tasks with at most 5% of outliers, $n = 25$ real inputs and $p = 1000$ examples.

the pattern recognition literature [54]. The Iris data set contains 3 classes of 50 examples referring to the three types of iris plants (virginica, versicolor and setosa). Four attributes are considered: the length and width of the sepals and petals. The first class is linearly separable from the other two but the second class is not linearly separable from the third one. In the latter case, the minimum number of errors is known to be equal to 1. Thus we have three tasks with $n = 4$ inputs and $p = 100$ examples each. The two linearly separable tasks are easy and all versions of the TPA or PPA give a zero error weight vector in just a few cycles. The third task is more interesting since it is nonlinearly separable and the minimum number of errors is known. Figure 5.9 shows the performances of the various algorithms for separating the second and the third classes. The focus is on the number of times an optimal solution is found over 100 runs when starting from different initial weight vectors. Both PPAs and the asymmetric TPA find an optimal weight vector in more than

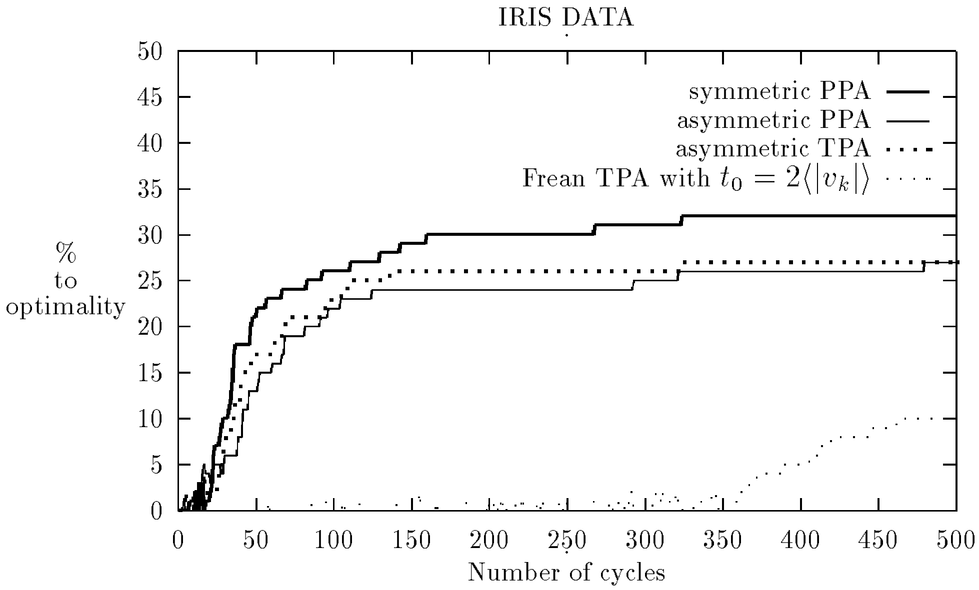


Figure 5.9: Iris data set with $n = 4$ attributes and $p = 100$ examples (second versus third classes). Average fraction of the time an optimal weight vector making a single error is find in the given number of cycles. Average over 100 runs of the algorithms starting with different initial weight vectors.

25 (and up to 32) out of 100 independent trials. Furthermore, they converge very rapidly to an average number of errors smaller than 2. In particular, the symmetric PPA achieves the value 1.84 ± 0.72. Burgess TPA version never reached an optimal solution in 100 runs of 500 cycles. The average number of errors decreases very slowly to 2.1 ± 0.1. Frean TPA

with an initial temperature $t_0 = 2\langle|v^k|\rangle$ finds an optimal solution in 1 out of 10 trials but its final average value of error is larger than 25.1 ± 24.05.

The above results show that our adaptive variants of the TPA converge much faster than the competing ones while providing at least as good solutions. Furthermore, they indicate that the probabilistic approach leads to comparable solutions.

5.7 Concluding remarks

We have first revisited the thermal perceptron algorithm that is extensively used in various methods for constructing and training multilayer networks. We provided an interpretation in terms of error function minimization and presented simple modifications that make the TPA adaptive as well as more efficient for linearly and nonlinearly separable tasks.

Then we have proposed probabilistic versions of the TPA that have two main advantages. First, they are guaranteed to converge and, second, they require lower precision computations. Therefore they are good candidates for harware implementation. The probabilistic approach can in fact be easily adapted to the case with discrete weights. Our idea has been successfully applied to perceptrons with ternary weights in [126]. It can also be used to extend the Directed Drift algorithm [162] for perceptrons with bipolar weights in order to deal with nonlinearly separable tasks.

As previously mentioned, the interest in this class of algorithms goes beyond the framework of artificial neural networks. Since they are simple variants of relaxation procedures, they can be used to determine near-optimal solutions of any possibly infeasible system of linear inequalities in the sense of MAX FLS or MIN ULR.

Finally, it is worth noting that the thermal and probabilistic approaches can be extended to train multilayer networks with a fixed topology. Instead of focusing on the least mean squared error, we can consider the function obtained by adding the perceptron criterion corresponding to each output unit. This error function can be minimized using a stochastic gradient technique similar to backpropagation. This is particularly attractive for classification problems where the classes are generally coded by the output units using a unary code and classification errors correspond to incorrect outputs. As in the perceptron case, the idea is to use the annealing process in order to minimize the number of incorrect outputs. The preliminary tests that we have been carrying out so far are promising.

Chapter 6

Designing feedforward networks

This chapter is devoted to algorithms for designing multilayer networks of linear threshold units. The focus is on constructive strategies that allocate units incrementally and train them using perceptron-like procedures. Section 6.1 briefly recalls the motivations and goals of network design. In section 6.2 we discuss the computational limitations on constructive strategies. In the following sections, three methods are proposed for designing different types of feedforward networks with binary or real inputs. The SNEAK algorithm described in section 6.3 builds parity machines. The SHIFT method constructs networks with a single hidden layer while the PTI algorithm may create an arbitrary number of hidden layers. Numerical results for tasks with binary and real-valued inputs are reported and discussed in section 6.6. The emphasis is on the size of the resulting networks and on their generalization properties. Finally, several general remarks and extensions are mentioned in section 6.7.

6.1 Network design

As we have seen in chapter 4, there is a strong interest in algorithms that do not only tune the weights of the network but also optimize its topology, in other words, in methods that design rather than just train the network. Such algorithms have two important advantages with respect to conventional methods like the commonly used backpropagation variants. First, the network topology is determined during the learning process and does not need to be guessed in advance, thus avoiding the very time-consuming trial-and-error procedure. Second, they allow to partially circumvent the inherent complexity of the problem of training networks with a fixed topology. In principle, algorithms that vary the weights as well as the topology can design networks that perform any given task [30]. This is in contrast with conventional training algorithms that are not guaranteed to converge to an appropriate set of weights, even if it exists, because the error function has typically many local minima.

Since compact networks tend to have better generalization properties, the design problem is a major issue in neural computation. The tradeoff is clearly that smaller networks are less powerful in terms of function representation. Given the complexity results discussed in section 4.4, the best we can do is to devise efficient heuristics with a good average-case behavior.

Two opposite approaches can be adopted to design compact networks. Either one chooses a network containing too many units and tries to remove redundant components, or one starts with a too small network and add new units as needed. While the first approach has been mainly applied to the case of continuous units [164], several methods for constructing networks of linear threshold units have been proposed in [61, 129, 124, 121]. In this chapter we focus on the latter approach.

Constructive algorithms start with a network that is not powerful enough to perform the given task T and allocate new units until it provides the correct output for all the input vectors of T. Most of them are guaranteed to produce in polynomial time a network that performs correctly any given task. Unlike the very time-consuming conventional methods, they require a small computational burden and involve local computations. This makes them attractive from a hardware implementation point of view. However, since units are allocated and trained sequentially, parallelism is not exploited during the learning phase.

In recent years, constructive methods have been applied to several practical problems [155, 42, 121, 123], but more effective procedures constructing more compact networks are needed to deal with large real-world applications.

Although all algorithms will be described for binary tasks, they can be easily extended to deal with real inputs using the stereographic projection as a preprocessing technique [148]. This issue will be discussed in section 6.6. For simplicity, we will adopt the $\{0, 1\}$ representation of binary values.

6.2 Limitations on constructive strategies

In spite of their particularities, all constructive methods are based on a *greedy* strategy. The global design problem is subdivided into a sequence of local training problems concerning single units. Starting with a network composed of a single unit, one (or a few) unit(s) is (are) added at each step and its (their) subtask(s) is (are) defined so as to correct errors made by the current network. In order to construct compact networks, the typical local goal is to train new units so that they make as few errors as possible on the corresponding subtasks. Clearly, the size of the resulting network depends on the global strategy as well as on the effectiveness of the method used locally.

Constructive strategies fall into two categories. In some algorithms, such as UPSTART [61], the first unit is the future output unit and new hidden units are interpolated between the input and output layers. In other algorithms, such as TILING [129], new units and layers are created downstream of the existing ones so that the network is extended outward. The first category includes *neural trees* that use linear threshold units for making decisions at each node of a binary tree [74, 155]. Procedures that construct networks with a single hidden layer and a predefined output function belong to the same category [62]. The SNEAK algorithm presented in section 4 builds *parity machines* whose output unit computes the parity function [131]. Two constructive algorithms that solve a sequence of linear programs have also been proposed. While the first one is restricted to simple halfspace intersections and neural decision lists [123], the Multisurface method can deal with any type of task with binary outputs [121].

In this chapter, we focus on strategies in which simple perceptron-like procedures are used to train the newly allocated units. Unlike the Linear Programming based methods that require complex optimization solvers, such algorithms are reasonable candidates for hardware implementations.

Constructive strategies share two kinds of limitations. The first one pertains to the hardness of training single units. As proven in chapter 4, maximizing consistency and minimizing the number of errors are not only hard to solve optimally but also to approximate. The situation becomes even harder when only a single type of error is allowed, like in the methods proposed in [124, 123]. This gives rise to the important question of whether the training problem (maximum consistency) is polynomially solvable for units that are restricted to perform a certain subclass of linear threshold functions. Of course, we are only interested in powerful enough subclasses[1] such that the resulting feedforward networks can still represent any boolean function. Unfortunately, it seems that no such class exists. For the usual subclasses like conjunctions, disjunctions and monomials the

[1] Although the maximum consistency problem can be solved in polynomial type for symmetric functions, the resulting networks can only perform symmetric functions.

maximum consistency is NP-hard and for linear threshold functions with binary weights and majority functions even the consistency question is known to be NP-hard. To our knowledge the question is, however, still open for parity functions.

The second limitation is inherent to the greedy approach. Breaking down the network design problem into that of training a sequence of single units is certainly attractive. However, even if we assume to find an optimum weight vector for each single unit, constructive strategies are not guaranteed to lead to minimum size networks. Clearly, these two limitations are related and they account for the hardness of the overall network design problem.

In the remainder of this section, we discuss these problems for the UPSTART and TILING algorithms that have been shown to compare favorably with respect to methods of the same category.

The UPSTART algorithm has been devised for constructing networks with a tree-like architecture and a single output unit that is directly connected to the input layer [61]. Starting with a network having no hidden units, the output unit is trained using a perceptron-like algorithm so as to minimize the number of errors on the given task T_0. Let z^k denote the output value for the input vector $\mathbf{a}^k$. If T_0 is nonlinearly separable, a number of errors will be left. Misclassifications may be of two different types: either the output is *wrongly-on* (i.e. $z^k = 1$ while the desired output is $b^k = 0$) or it is *wrongly-off* (i.e. $z^k = 0$ while $b^k = 1$). Let $\tilde{A}^+$, $\tilde{A}^-$, A^+ and A^- denote the sets of input vectors $\mathbf{a}^k$ that are respectively wrongly-on, wrongly-off, correctly-on or correctly-off. If $|\tilde{A}^+| \neq 0$, a new hidden unit is added between the input and output layers. The purpose of this auxiliary unit is to correct the wrongly-on errors of the output unit and its task is defined as $T_1 = \{(\mathbf{a}^k, 1) \mid \mathbf{a}^k \in \tilde{A}^+,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid \mathbf{a}^k \in A^+ \vee \tilde{A}^-,\ 1 \leq k \leq p\}$. Similarly, when $|\tilde{A}^-| \neq 0$ an additional unit is created and it is assigned the task $T_2 = \{(\mathbf{a}^k, 1) \mid \mathbf{a}^k \in \tilde{A}^-,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid \mathbf{a}^k \in \tilde{A}^+ \vee A^-,\ 1 \leq k \leq p\}$. If the first (second) auxiliary unit performs its task T_1 (T_2) correctly and is connected to the output unit with a large enough negative (positive) weight, it corrects all wrongly-on (wrongly-off) input vectors. In case T_i, $1 \leq i \leq 2$, is nonlinearly separable, a simple greedy strategy is adopted. One tries to correct as many mistakes of the output unit as possible by minimizing the number of errors on T_i. Whenever needed new auxiliary units are allocated to correct the errors of the previous "parent" unit i. This recursive procedure is applied until all the hidden units perform the corresponding tasks correctly. The resulting network has a hierarchical topology. Convergence is guaranteed after adding at most p hidden units because auxiliary units always make fewer mistakes than their parent units.

Notice that if the input vectors $\mathbf{a}^k \in \tilde{A}^-$ and $\mathbf{a}^k \in \tilde{A}^+$ were not included in the subtasks T_1 and T_2 respectively, UPSTART would divide the input space like a neural tree [74]. In this case, the boolean function to traverse the tree is built into the network. Alternatively, the units of the decision tree may be inserted in a single hidden layer and the function to traverse it can be represented explicitly using 2 additional layers [154]. In order to be able to squash the networks produced by UPSTART into a single hidden layer, it has been proposed to keep all $\mathbf{a}^k \in A^-$ in T_1 and all $\mathbf{a}^k \in A^+$ in T_2 [61].

In spite of its convergence guarantee, minimizing the number of errors made by each unit does not necessarily result in an optimum network.

Proposition 6.2.1 *Given any binary task T, the* UPSTART *algorithm for constructing feedforward networks with direct input-to-output connections is not guaranteed to lead to a minimum size network consistent with T.*

PROOF We give an example of a binary task for which the greedy strategy builds a network that is not of minimum size. Neglecting the difficulty of training single units, we assume that an optimal weight vector can be found for each unit. A simple example is provided by the *parity* task in which a vector $\mathbf{a}^k \in \mathbb{B}^n$ is assigned a target $b^k = 1$ if it contains an odd number of 1 components and a target $b^k = 0$ otherwise. Consider the parity task with $n = 3$ and $p = 8$ represented in Figure 6.1. If the number of errors made by the output unit is minimized, one obtains a weight vector corresponding to a hyperplane with 2 mistakes. The hyperplane H_1 shown in (a) makes one wrongly-on and one wrongly-off errors and 2 additional hidden units are needed to perform the task correctly. However, it is easily verified that if the output unit corresponds to hyperplane H_1 shown in (b), which makes 3 errors, a single hidden unit associated to H_2 and connected with an appropriate positive weight to the output unit suffices to correct all mistakes. The positive weight must be large enough to correct the wrongly-off input vectors but not too large so that it does not corrupt the output for the input vector with $b^k = 0$ lying on the positive side of H_2. $\square$

The parity task of size n has often been cited as a hard problem because input vectors which differ by a single component require different outputs. As well known, this predicate of order n can be implemented by a two-layer network with n hidden units and no connection to the input layer [144]. The above example suggests that fewer hidden units are sufficient when direct input-to-output connections are allowed. In fact, it is readily

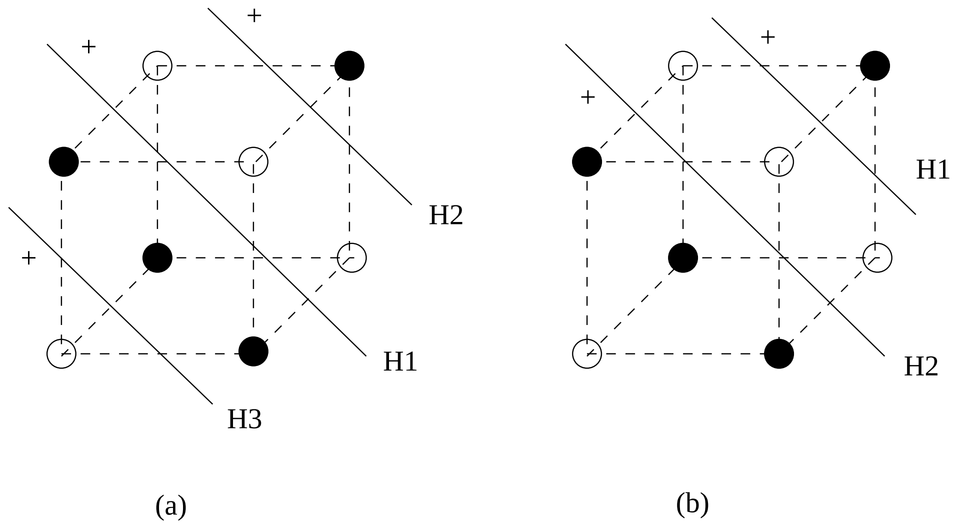


Figure 6.1: Two alternative solutions for the parity task with $n = 3$ and $p = 8$ when there are direct input-to-output connections. The task is represented on the 3-dimensional hypercube, black points denote 1 targets and white ones 0 targets. The positive side of each separating hyperplane is indicated by $+$. (a) The output unit corresponding to hyperplane H_1 makes a minimum number of errors (namely 2) and 2 additional units are needed. The hidden unit associated to H_2 corrects the wrongly-on error and that associated to H_3 corrects the wrongly-off one. (b) The output unit corresponding to hyperplane H_1 does not make a minimum number of errors but a single hidden unit associated to H_2 is sufficient to correct the 3 mistakes.

verified that a network with $\lceil \frac{n+1}{2} \rceil - 1$ hidden units and direct input-to-output connections is able to perform the parity task of size n. Note that this solution has nearly half as many hidden units as the one mentioned in [62]. To the best of our knowledge, this very compact solution has not yet been discovered.

Proposition 6.2.1 shows that even if the number of errors made by a perceptron could be minimized in polynomial time, there would be no guarantee of designing minimum size networks.

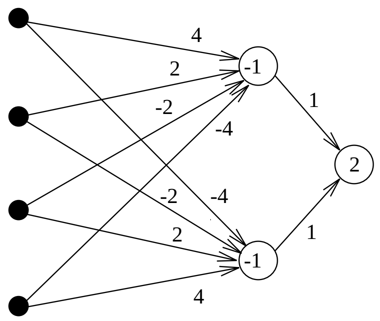
Unlike Upstart, the Tiling algorithm builds the network layer by layer until all associations of the task are performed correctly [129]. Suppose l layers have been created so far. When an input vector $\mathbf{a}^k$ is presented to the network, it provides a vector $\tilde{\mathbf{a}}_l^k$ which corresponds to the output of the units in the l-th layer. For any $1 \leq k \leq p$, $\tilde{\mathbf{a}}_l^k$ is referred to as the *internal representation* of $\mathbf{a}^k$ on the l-th layer. The set of all input vectors $\mathbf{a}^k$ that have the same internal representation are said to belong to the same *class*. When building the $(l+1)$-th layer, the input vectors with internal representation $\tilde{\mathbf{a}}_l^k$ need to be mapped onto their targets b^k. In each layer there are two different types of units : the *master* unit and the *ancillary* ones. The first unit added on a new layer is the master whose goal is to correctly classify as many input vectors $\mathbf{a}^k$ as possible. If errors are left, ancillary units are added to complete the current layer; their objective is quite different. Clearly, a necessary condition to be able to perform the task is that all input vectors $\mathbf{a}^k$ of a same class have the same target value b^k. A class of vectors that does not satisfy that condition is said to be *unfaithful*. Ancillary units are added until all classes are faithful. A divide and conquer strategy has been proposed to build faithful classes. Starting with the smallest unfaithful class, a new unit is added to the current layer and is trained to map $\tilde{\mathbf{a}}_l^k$ onto b^k for all internal representations $\tilde{\mathbf{a}}_l^k$ associated to that class. If this mapping is performed correctly, the class is faithful and we can try to use the same unit to make also the next smallest class faithful. If it is not, the previous class will be divided into two smaller classes. The process always terminates because each new unit divides at least one unfaithful class into two smaller classes. From a global point of view, the algorithm converges because each new layer reduces the number of misclassified input vectors $\mathbf{a}^k$ by at least one.

Since Tiling is also based on a greedy strategy, it shares the same limitation of Upstart.

Proposition 6.2.2 *Given any binary task T, the* Tiling *algorithm for constructing multilayer networks is not guaranteed to lead to a minimum size network consistent with T.*

Proof As in the proof of proposition 6.2.1, we assume that single units are trained optimally and we exhibit a case for which Tiling is sub-optimal. A simple example is provided by the *symmetry* task in which a vector $\mathbf{a}^k \in \mathbb{B}^n$ with an even number of components n is assigned a target $b^k = 1$ if it is symmetric about its center and a target $b^k = 0$ otherwise. For any size n, this task can be performed by a two-layer network with only 2 hidden units [145]. Consider the symmetry task with $n = 4$ and $p = 16$. In the minimum solution shown in Figure 6.2 (a), both hidden units are off when the input vector is symmetric and at least one of them is on for every nonsymmetric input vectors. Note that each hidden unit makes 8 mistakes on the overall symmetry task.

If single units are trained optimally, Tiling constructs a two-layer network with 4 hidden units. Although the master unit makes only 3 mistakes, three additional ancillary units are needed in order to make all classes faithful. For the optimal weight vector of the master unit given in Figure 6.2 (b), the smallest unfaithful class can be broken down using a single ancillary unit. However, two additional ancillary units are required to make the largest one faithful. □



Weight vector making 3 errors: w=(-1,-3,2,-3)

Smallest unfaithful class (output 1): (0,0,0,0) (1,0,1,0) (0,0,1,0)

Largest unfaithful class (output 0): (0,1,0,1) (1,1,1,1) and all other 11 negative examples

(a) (b)

Figure 6.2: Symmetry task with $n = 4$ and $p = 16$. (a) Minimum size network performing the symmetry task. The corresponding weights are denoted besides each connection and the thresholds inside the corresponding units. Both hidden units make 8 errors on the overall task. (b) In TILING, minimizing the number of errors made by each single unit, leads to a network with 4 hidden units. An optimal weight vector for the master unit is given as well as the two corresponding unfaithful classes. The smallest one contains 3 input vectors and the largest one 13 input vectors. Three ancillary units are needed to make both of them faithful.

Propositions 6.2.1 and 6.2.2 depend of course on the specific constructive methods. Similar counter-examples can however be exhibited for the other greedy strategies. This is in particular true for those that we will present. Although more sophisticated methods may be developed, the greedy approaches are attractive because of their simplicity, locality and relatively small computational requirements. In any case, the complexity results discussed in section 4.4 imply that any design method of whatever level of sophistication faces computational limitations.

Due to the kind of limitations proper to the greedy strategies, it does not seem necessary, and even desirable, to try to solve the local problems optimally. Using effective but sub-optimal training methods, might even turn out to be advantageous from the overall network point of view. The tradeoff is of course not clear. However, it might not be worthwhile to use complex methods such as Linear Programming for tackling the local problems.

In the next sections, we present the three algorithms that we devised to construct different types of feedforward networks.

6.3 The SNEAK algorithm

Before presenting the PTI and SHIFT algorithms for designing arbitrary networks of linear threshold units, we describe a method for constructing parity machines. This algorithm, which we refer to as SNEAK, has been independently developed in [125]. A variant of this procedure will be used in the PTI method that we will propose in section 6.4.

A parity machine (PM) is a feedforward network with a single hidden layer of threshold units and an output unit computing the parity function (XOR). The hidden units are connected to all the inputs and the output unit is only connected to the hidden layer. Since the parity function with n variables can be implemented by a two layer network with n hidden units, a PM may be considered as a three-layer network of threshold units. This type of architecture has been studied in [131], where an algorithm was proposed for training PMs with a fixed topology, and their information storage capacity has been thoroughly investigated in [27, 28].

It is well known that PMs can implement any boolean function provided enough hidden units are available. Two-layer networks with an output unit computing other predefined boolean functions (like majorities) are also able to represent any boolean function. However, numerical evidences indicate that PMs have a larger storage capacity than networks with other fixed output function [131].

According to the first constructive strategy line, we assume that the output function is fixed and that the appropriate number of hidden units has to be determined.

Before describing the algorithm, we address the computational complexity of designing this particular type of feedforward networks.

Proposition 6.3.1 *Given any binary task T, it is* NP*-complete to decide whether T can be performed by a PM with two hidden units.*

PROOF Blum and Rivest proved that the question is NP-complete for PMs with ternary inputs in $\{-1, 0, 1\}$ [40]. The technique used in the proof of theorem 3.1.4 allows to extend this result to PMs with only bipolar inputs. $\square$

Thus, it is intractable to design near-optimum PMs that are consistent with any T and have strictly less than $\frac{3}{2}$ times as many hidden units as the minimum size ones.

The SNEAK algorithm constructs a PM by allocating one new hidden unit at each step until the current unit makes no error on the assigned task. We start with a single hidden unit. Obviously such a network can only perform linearly separable tasks. The hidden unit is trained on the given task T_0 so as to minimize the number of misclassifications. If T_0 is nonlinearly separable or if the perceptron variant used does not find an optimum weight vector with zero errors, an additional hidden unit is generated. The role of the new unit is to indicate (SNEAK) which input vectors $\mathbf{a}^k$ of T_0 were misclassified by the previous hidden unit. In other words, the new unit should be on for all $\mathbf{a}^k$ that were previously misclassified and off for all the others.

Let $T_i \subset \mathbb{B}^n \times \mathbb{B}$ denote the task assigned to the i-th hidden unit. Suppose that i units have been created and that the i-th one makes some errors on T_i. Then a new $(i+1)$-th hidden unit is added and its task T_{i+1} is defined as follows:

$$T_{i+1} = \{(\mathbf{a}^k, b_{i+1}^k) \mid (\mathbf{a}^k, b^k) \in T_0 \ \wedge \ b_{i+1}^k = \mathrm{XOR}(y_i^k, b_i^k), \ 1 \leq k \leq p\},$$

where y_i^k denotes the output of the i-th unit for input $\mathbf{a}^k$. If errors are left after applying an efficient TPA variant, a new hidden unit is allocated.

While the first hidden unit attempts to perform T_0 correctly, the goal of the i-th one, $i \geq 2$, is to try to correct the output for the input vectors that were misclassified by the previous $(i-1)$-th unit. Since the output for several $\mathbf{a}^k$ that were correctly classified may be corrupted, one must try to correct them with the next hidden unit.

This simple iterative procedure is guaranteed to converge to a PM that performs the task.

Theorem 6.3.2 *For any binary task T, the* SNEAK *algorithm constructs a PM that is consistent with T.*

PROOF First we verify that the algorithm terminates after creating at most p hidden units. Consider the set $A_i = \{\mathbf{a}^k \mid (\mathbf{a}^k, 1) \in T_i, \ 1 \leq k \leq p\}$. By definition of T_i, its cardinality $|A_i|$ is equal to the number of $\mathbf{a}^k$ that are incorrectly classified by the previous $(i-1)$-th unit. If the weights and the threshold of the new i-th unit are chosen so that it is on for one $\mathbf{a}^k \in A_i$ and off for all the others, the i-th unit misclassifies at most $|A_i| - 1$ input vectors of T_{i+1}. Hence, using any algorithm for minimizing the number of errors that provides a solution that is at least as good as the above-mentioned trivial solution, we are guaranteed that $|A_i| \leq |A_{i-1}| - 1$.

To show that the resulting PM is consistent with the task T, we verify that a PM with h hidden units correctly classifies an input vector $\mathbf{a}^k$ if and only if the h-th unit provides the right input according to its task T_h. This can be simply done by induction on h. The claim is obviously true for $h = 1$. Suppose it holds for i. Since the task T_{i+1} assigned to the $(i+1)$-th unit is obtained by making a XOR of the targets in T_i and the corresponding outputs given by the i-th unit, this is also valid for $i+1$. Thus, if the algorithm terminates after having allocated h hidden units, the h-th unit makes no errors on T_h and the overall PM is consistent with T. $\square$

In practice, when applying any TPA variant, one should check whether the weights and threshold given by the local training algorithm make fewer mistakes than the trivial solution. If it is not the case, the number of errors does not strictly decrease when the new hidden unit is added and the trivial solution must be selected. As we shall see in section 6.6, the number of units that are allocated is in fact much smaller than p.

As UPSTART and TILING, the SNEAK algorithm does not necessarily produce an optimum network. A counter-example is provided by the symmetry task with $n = 4$ and $p = 16$. In this case, the size of the resulting PM may even depend on the weight vectors that have been chosen among those making a minimum number of errors.

Since PMs can be represented as three-layer networks of threshold units, redundant units in the second hidden layer which computes the XOR function can be removed. The simplest way to proceed is to consider the second layer units sequentially and delete those leaving all internal representations $\tilde{\mathbf{a}}^k$ faithful [125]. Ideally, one would like to select among all XOR symmetries that perform T the one that requires the smallest number of units in its network representation. Although exhaustive procedure is computationally prohibitive, we have checked that an optimization heuristic technique like tabu search [69, 71] can be easily adapted.

6.4 The PTI algorithm

The second algorithm we propose is named PTI (Partial Target Inversion) and it constructs the network layer by layer like TILING.

Suppose that l layers have been built so far. The internal representation of all input vectors in a class c is denoted by $\hat{\mathbf{a}}_l^c$ and is called its *prototype*. The following simple geometrical interpretation provides some insight into the faithfulness condition. Since each unit of the $(l+1)$-th layer divides the space spanned by the prototypes $\hat{\mathbf{a}}_l^c$ into two halfspaces, the layer determines a set of regions corresponding to the intersection of all those halfspaces. In such a representation, the faithful classes correspond to the regions in which all prototypes $\hat{\mathbf{a}}_l^c$ have the same target.

In PTI, hidden units are allocated in each layer using a SNEAK-like strategy. While the first unit in the $(l+1)$-th layer attempts to map the prototypes $\hat{\mathbf{a}}_l^c$ of each class onto the corresponding targets, the following ones try to indicate which input vectors were misclassified by the previous hidden units. In this case, however, only the prototypes $\hat{\mathbf{a}}_l^c$ belonging to unfaithful classes are included in the task of the new unit.

Let T_i^{l+1} denote the task assigned to the i-th unit of layer $l+1$. Assume that i units have been inserted so far in this layer. The task T_{i+1}^{l+1} assigned to the $(i+1)$-th unit is defined as in SNEAK by comparing the target value in T_i^{l+1} and the actual output value of the i-th unit. Since the role of new hidden units is to ensure faithful representations, T_{i+1}^{l+1} does not contain the prototypes $\hat{\mathbf{a}}_l^c$ of T_i^{l+1} that belong to faithful classes. With respect to T_i^{l+1}, this amounts to assign the *same* target to the prototypes $\hat{\mathbf{a}}_l^c$ for which the i-th unit is off and the *opposite* target to those for which it is on. The rationale is that, when trying to break several unfaithful classes simultaneously, it may be worthwhile to flip the class membership of the prototypes in several unfaithful classes. Indeed, the particular targets are not important, the only requirement is that prototypes corresponding to different b^k trigger a different output for at least one unit in the current layer. By using a SNEAK-like procedure, one flips the class membership of the prototypes lying on the positive side of the hyperplane associated to the last allocated unit. The current layer is obviously achieved when the current unit perform its task correctly.

Figure 6.3 illustrates the constructive process for the parity task with $n = 3$ inputs and $p = 8$ examples.

PTI always terminates and provides a multilayer network that is consistent with any given task.

Lemma 6.4.1 *Consider a network with l layers and suppose that $\tilde{\mathbf{a}}_l^k$ is a faithful internal representation of $\mathbf{a}^k$ for all $1 \leq k \leq p$. The* PTI *procedure for adding hidden units in a new $(l+1)$-th layer terminates and ensures that the $\tilde{\mathbf{a}}_{l+1}^k$ are faithful representations of the $\mathbf{a}^k$.*

PROOF According to SNEAK finite convergence guarantee, the PTI procedure for adding units in the current layer terminates after allocating a number of units that is at most equal to the number of prototypes $\hat{\mathbf{a}}_l^c$ which have a 1 target in T_1^{l+1}. Since the prototypes belonging to faithful classes are omitted, the number of allocated units is smaller in this case.

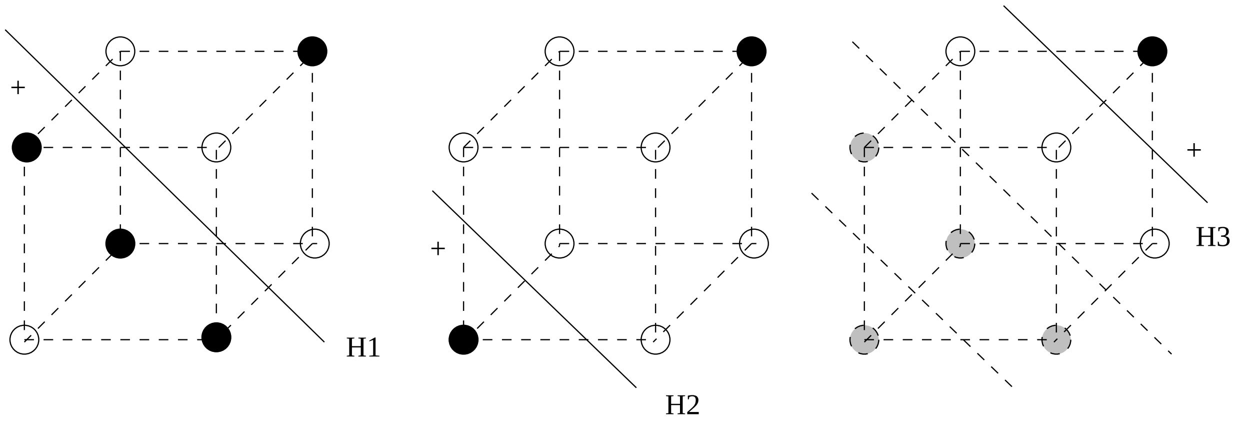


Figure 6.3: Tasks of the 3 hidden units generated by PTI for the parity problem with n=3 and p=8. (a) The first unit in the hidden layer makes a minimum number of errors (namely 2) on the original task. (b) The second hidden unit must isolate the 2 examples that are misclassified by the first unit from the other ones but it cannot achieve this subtask completely. (c) The third hidden unit must separate the input vector that is misclassified by the second unit from the white ones. The input vectors represented in grey have been discarded because they belong to faithful classes.

In order to show that the $\tilde{\mathbf{a}}_{l+1}^k$ are faithful representations of the input vectors $\mathbf{a}^k$, it suffices to check that the representations of the prototypes $\hat{\mathbf{a}}_l^c$ on the $(l+1)$-th layer are faithful with respect to the task T_1^{l+1}. Let $\hat{\mathbf{a}}_l^{c_1}$ and $\hat{\mathbf{a}}_l^{c_2}$ be any pair of prototypes occurring in T_1^{l+1} with opposite targets. It is easily verified that there is always at least one unit in the $(l+1)$-th layer that gives a different output for each of them. Suppose that the $(l+1)$-th layer contains h units. Only two cases can arise:
(i) $\hat{\mathbf{a}}_l^{c_1}$ and $\hat{\mathbf{a}}_l^{c_2}$ have different targets in T_h^{l+1}. Since this task is performed correctly, they have different representations on the l-th layer.
(ii) $\hat{\mathbf{a}}_l^{c_1}$ and $\hat{\mathbf{a}}_l^{c_2}$ have the same target in T_h^{l+1}. By construction, this is only possible if at least one of the previous $h-1$ units gives an output 1 for one of the prototypes and an output 0 for the other one. □

Theorem 6.4.2 *For any given binary task T, the* PTI *algorithm constructs a feedforward network that is consistent with T.*

Proof The proof is similar to that for the Tiling algorithm [129]. However, instead of showing that the number of errors (misclassified vectors $\mathbf{a}^k$) decreases for each new master unit, we can verify that the total number of classes for which the first unit in a layer provides the wrong output decreases monotonically. Thus, rather than focusing on the total number of errors on the overall task, we consider the number of local errors made on the task assigned to the first unit in successive layers. □

Two fundamental differences with respect to TILING are worth noting. First, in PTI there is no distinction between *master* and *ancillary* units; the same local procedure is used to train both types of units. Second, TILING uses a divide and conquer strategy to construct the ancillary units, in which each new unit divides an unfaithful class of vectors into two smaller classes. In our approach, instead of breaking down the problem and considering only one unfaithful class at the time, all unfaithful classes of input vectors are considered simultaneously at each step. As we will see in section 6.6, this holistic approach turns out to be more efficient.

Not surprisingly, also the PTI is not guaranteed to yield minimum size networks.

6.5 The SHIFT algorithm

The third constructive algorithm we propose is named SHIFT and, unlike PTI, it builds networks with a single hidden layer. It starts with a network composed of a single unit (the output unit) that is trained on the given task $T_0 = \{(\mathbf{a}^k, b^k)\}_{1 \leq k \leq p}$. If T_0 is nonlinearly separable, new units are allocated in a single hidden layer until all associations are performed correctly. Let v^k denote the total input of the output unit when the vector $\mathbf{a}^k$ is presented to the network. For any input vector $\mathbf{a}^k$, the output unit is either wrongly-on, wrongly-off, correctly-on or correctly-off. Consider the effect of adding a new i-th unit to the hidden layer. For input vectors $\mathbf{a}^k$ such that the new unit takes output $y^k = 1$, the corresponding total input amounts to $v^k + \Delta_i$, where Δ_i denotes the value of its connection to the output unit. For $\mathbf{a}^k$ such that $y^k = 0$, v^k remains unchanged.

As in UPSTART, new units are in charge of correcting either wrongly-on or wrongly-off errors. However, here we do not proceed recursively. All units are added in a single layer and each of them corrects errors of the output unit. In order to correct wrongly-on mistakes, the total input v^k for the wrongly-on input vectors must be reduced while leaving v^k unchanged for the correctly-on ones. This might be achieved by adding a new unit providing an output 1 when $v^k > 0 \wedge b^k = 0$ and 0 when $v^k > 0 \wedge b^k = 1$. Clearly, a small enough weight $\Delta < 0$ should be selected. Since the corresponding task

$T = \{(\mathbf{a}^k, 1) \mid v^k > 0 \wedge b^k = 0,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid v^k > 0 \wedge b^k = 1,\ 1 \leq k \leq p\}$ may be nonlinearly separable, the new unit will be trained to perform it as well as possible. To maximize the number of wrongly-on errors corrected by the new unit, we can take $\Delta \leq -v_{\max}$, where $v_{\max} = max_k\{v^k \mid y_k = 1 \wedge b_k = 0\}$ is the largest total input of all $\mathbf{a}^k$ for which the output unit is wrongly-on and the newly added unit is on.

Wrongly-off errors are corrected in a similar way. The new unit is assigned the task $T = \{(\mathbf{a}^k, 1) \mid v^k \leq 0 \wedge b^k = 1,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid v^k \leq 0 \wedge b^k = 0,\ 1 \leq k \leq p\}$ and the connection weight to the output unit is taken as $\Delta = -v_{\min} + \varepsilon$, where $v_{\min} = min_k\{v^k \mid y_k = 0 \wedge b_k = 1\}$ and ε is a small positive value. The ε modification is due to the fact that a zero total input gives rise to a 0 output.

Depending on the most frequent type of errors, the new unit is assigned to correct wrongly-on or wrongly-off errors. The total inputs v^k are clearly updated every time a new hidden unit has been allocated.

This simple algorithm is guaranteed to yield a network that performs T_0 correctly.

Theorem 6.5.1 *For any given binary task T, the* SHIFT *algorithm constructs a two-layer network consistent with T.*

PROOF Consider a network with i units on the hidden layer. Let e_i^+ denote the number of wrongly-on errors, e_i^- the number of wrongly-off errors and $e_i = e_i^+ + e_i^-$ the total number of errors. It suffices to verify that $e_{i+1} < e_i$. We consider only the case where the $(i+1)$-th unit is used to correct wrongly-on errors. The wrongly-off case is dealt with similarly.

By definition, the task assigned to the $(i+1)$-th unit contains exactly e_i^+ associations of type $(\mathbf{a}^k, 1)$. Because of the trivial solution, the number of errors $\tilde{e}$ made by this unit is not greater than $e_i^+ - 1$.

Let v_i^k denote the total input of the output unit when i hidden units are present, y_i^k the output of the i-th unit for $\mathbf{a}^k$ and Δ the weight connecting that unit to the output unit. After the $(i+1)$-th unit has been allocated to correct wrongly-on errors, only two types of errors are left: wrongly-off errors, which were already present before adding the $(i+1)$-th unit, and errors that have been introduced because the $(i+1)$-th unit gives the wrong output with respect to T_{i+1}^l. In other words, $e_{i+1} = \tilde{e} + e_i^- \leq e_i^+ - 1 + e_i^- = e_i - 1$. $\square$

The behavior of the algorithm for the parity problem suggests some simple improvements. Let Q be an arbitrary vertex of the hypercube $\{0, 1\}^n$. All points that are at Hamming distance r from Q are said to be on the same *slice*, denoted by S_r. While S_0

consists only of Q and S_n only of $-Q$, slice r contains C_r^n vertices, i.e. all points that differ from Q in r components. In the parity problem, the input vectors $\mathbf{a}^k$ lying on the same slice have the same target while $\mathbf{a}^k$ on adjacent slices S_r, S_{r+1} have opposite targets. If $Q = (0, 0, .., 0)$, the hyperplane defined by $\sum_{i=0}^{n} x_i - r + \frac{1}{2} = 0$, separates the slices $S_0, ..., S_r$ from those $S_{r+1}, ..., S_n$.

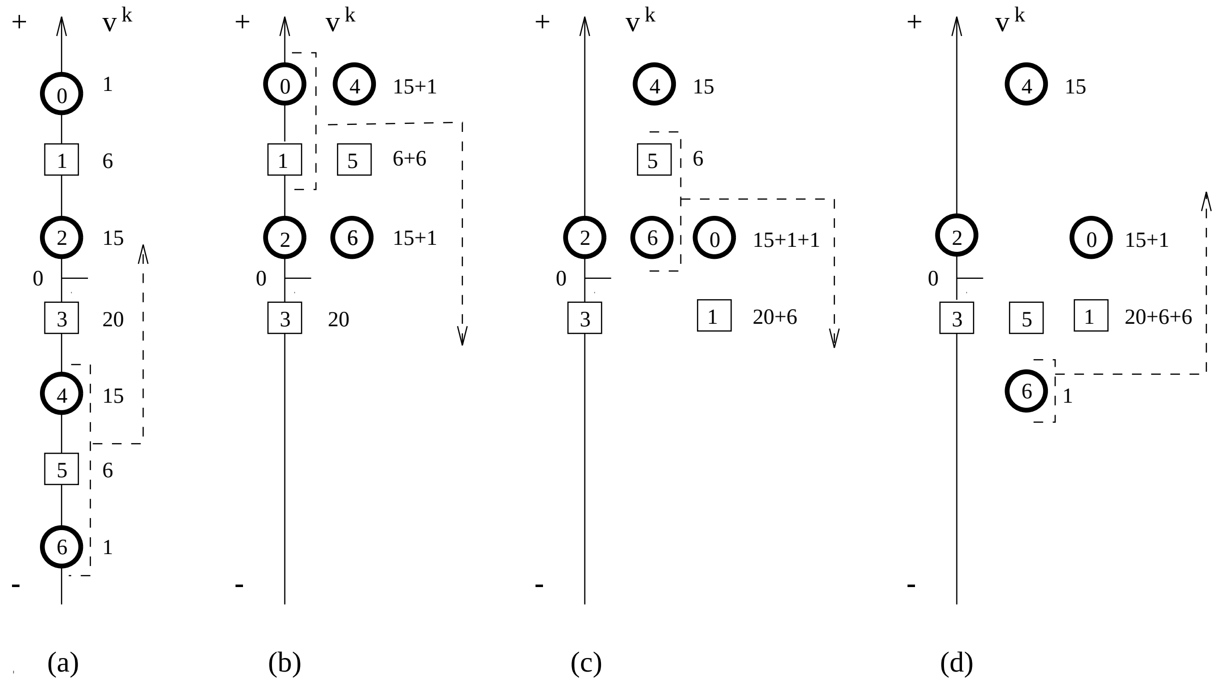


Figure 6.4: Parity problem with $n = 6$ and $p = 64$. In (a) the network is composed of a single unit, the output unit and there are 22 errors left. In (b) a unit has been added to correct wrongly-off errors and only 12 errors remain. (c) and (d) show the situation after 2 and 3 hidden units have been added. Since a single error remains in (d), the resulting network contains 4 hidden units.

Figures 6.4 and 6.5 illustrate how the algorithm works for the parity problem with $n = 6$ inputs and $p = 64$ examples. Input vectors $\mathbf{a}^k$ are placed along a vertical axis at a position determined by the corresponding total input value v^k. White squares represent input vectors with target $b^k = 0$ while solid circles represent those with target $b^k = 1$. Intuitively, we can consider that a hidden unit assigned to correct wrongly-on errors and performing its task correctly, is *shifting* all the white squares sitting above the origin below it, while leaving the black circles at the same position. Note that for the parity problem, instead of representing the 2^n associations of the task, the input vectors $\mathbf{a}^k$ are grouped into slices. The index and the cardinality of each slice are indicated respectively inside and beside the corresponding symbol.

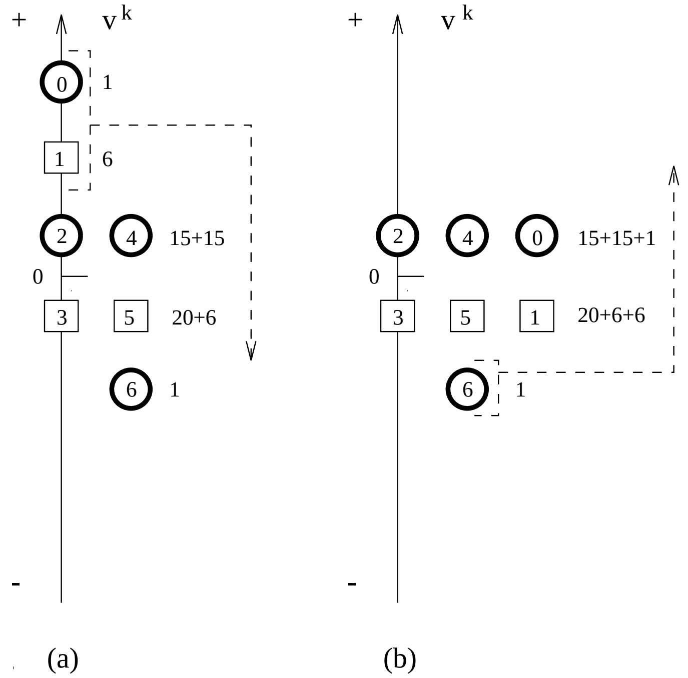


Figure 6.5: Parity problem with $n = 6$ and $p = 64$. Figure 6.4 (a) gives the initial situation with no hidden unit. In (a) a unit is added to correct wrongly-off errors and only 7 errors remain. In (b) only one error is left so that the resulting network contains 3 hidden units.

Although we pointed out in section 6.2 that this problem can be solved by a network with only three hidden units and direct input-to-output connections, Figure 6.4 shows that the present version of SHIFT requires at least four hidden units. The problem is due to the choice of the weights Δ connecting the new units to the output unit. However, there is a simple way of solving it. For a new unit assigned to correct wrongly-on errors, we consider among all total inputs v^k the positive ones for which the actual output of the new hidden unit is 1. These correspond to the input vectors for which the output unit was on and will be modified when the new hidden unit is added. If those v^k are sorted so that $v^i \leq v^j, \forall i < j$, choosing $\Delta = -\frac{1}{2}(v^{k+1} + v^k)$ makes the output unit off for all input vectors a^k with $1 \leq k \leq i$ and on for all a^k with $i < k \leq p$. Thus, considering the different k allows to select the value of Δ that minimizes the number of errors. A similar procedure is also applied in the wrongly-on case.

The information required in the previous modification can be used in a further improvement. Let $v_{\max}$ be the largest value v^k for which the output unit is wrongly-on. In order to correct any wrongly-on error, it suffices to reduce the corresponding total input v^k by an amount of at most $v_{\max}$. But if $\Delta \geq -v_{\max}$, all correctly-on input vectors with $v^k > v_{\max}$ remain correctly classified regardless of the output of the new unit. Therefore, the new unit can be assigned the task $T = \{(\mathbf{a}^k, 1) \mid v^k > 0 \wedge b^k = 0,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid 0 < v^k < v_{\max} \wedge b^k = 1,\ 1 \leq k \leq p\}$. The same situation arises when trying to correct wrongly-off errors. In this case, the corresponding task is $T = \{(\mathbf{a}^k, 1) \mid v^k \leq 0 \wedge b^k = 1,\ 1 \leq k \leq p\} \cup \{(\mathbf{a}^k, 0) \mid 0 \geq v^k \geq v_{\min} \wedge b^k = 0,\ 1 \leq k \leq p\}$.

Finally, it is worth noting that direct connections between the input units and the output unit can be eliminated. Depending on the threshold value, all initial errors will then be either wrongly-on or wrongly-off. Since allowing direct input-to-output connection sensibly increases the representation power [156], simpler networks without direct connections might be preferred.

6.6 Experimental results

Three test problems have been selected in order to compare the performances of the various constructive algorithms according to the two important criteria: network size and generalization abilities. Experiments have also been carried out on a real-world problem with real-valued inputs.

6.6.1 Network size

First we focus on the network size. Therefore we consider random binary tasks where all 2^n possible input vectors are assigned a target 1 or 0 with equal probability. This kind of tasks allows to compare the ability of the different algorithms to construct compact networks. As previously seen, finding a compact representation is not only of interest by itself but it has also a strong impact on the network generalization properties. The size can be measured in terms of the number of units or the number of nonzero weights. For networks with more than two layers, the number of layers is also important.

Figures 6.6, 6.7 and 6.8 report the average number of units, weights and layers for random tasks with various input sizes n. All results are averages on 25 different random tasks. Standard deviations are also shown.

Although the networks constructed by PTI have a considerably smaller number of units than those provided by TILING, SNEAK and UPSTART, they are larger than those produced by SHIFT. As shown in Figure 6.7, the situation is slightly different in terms of weights. The networks built by SHIFT contain definitely less weights than those obtained with PTI. This can be explained by the fact that the first layers generated by PTI contain a number of units which is in general much larger than the number of inputs n.

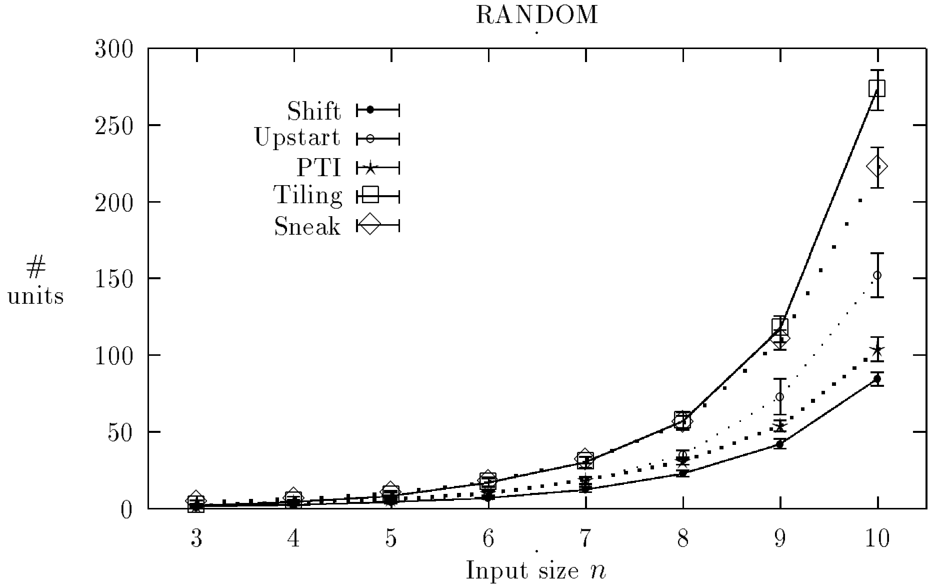


Figure 6.6: Random tasks with $p = 2^n$ examples for various input sizes n. Average number of units in the networks constructed by the different algorithms. Averages are over 25 different tasks.

The SHIFT algorithm turns out to build networks containing less hidden units and less weights than those produced by UPSTART. Moreover, they have a single hidden layer instead of a tree-like topology with a very large number of layers. Not surprisingly, the UPSTART variant which allows to contract the tree in a single hidden layer performs sensibly worse than the original method. As shown in Figure 6.9, keeping all input vectors of the original tasks in the subsequent subtasks leads to networks that are substantially larger than those obtained without task reduction.

SNEAK results correspond to the basic version without pruning phase. Using the simple pruning procedure, yields networks that are smaller than those produced by TILING but still considerably larger than those built by UPSTART [125].

Is is worth noting that, as shown in Figure 6.9, the improvements introduced in the

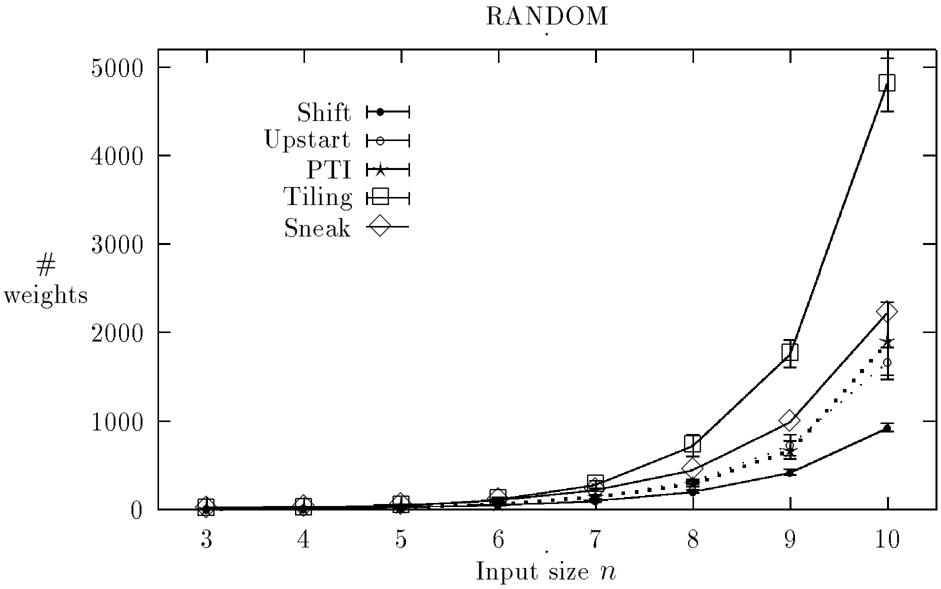


Figure 6.7: Average number of weights over 25 different random tasks. For SNEAK the weights involved in the network representation of the parity function are neglected.

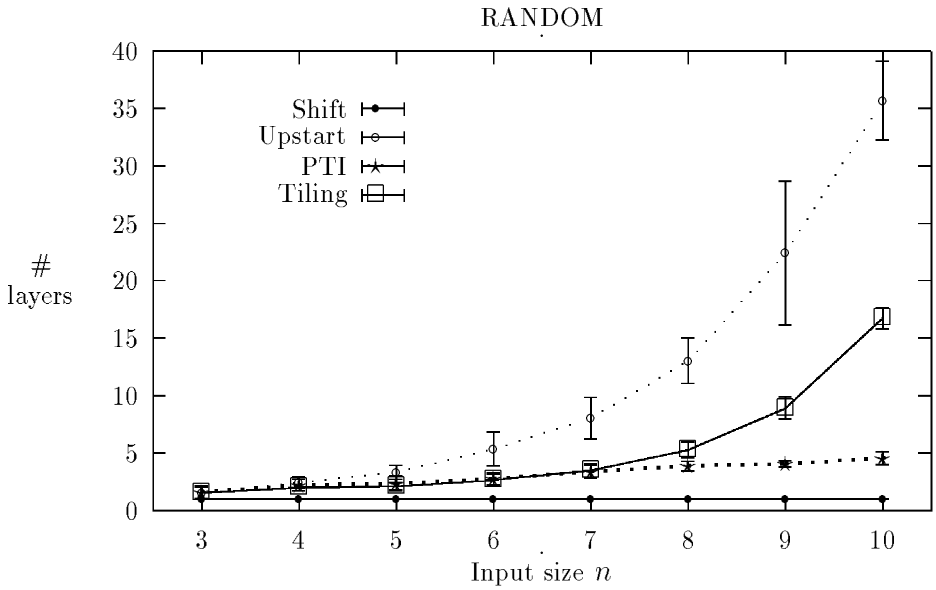


Figure 6.8: Average number of layers over 25 different random tasks. The SHIFT algorithm constructs networks with a single hidden layer.

SHIFT algorithm are interesting in terms of network compactness even for random tasks

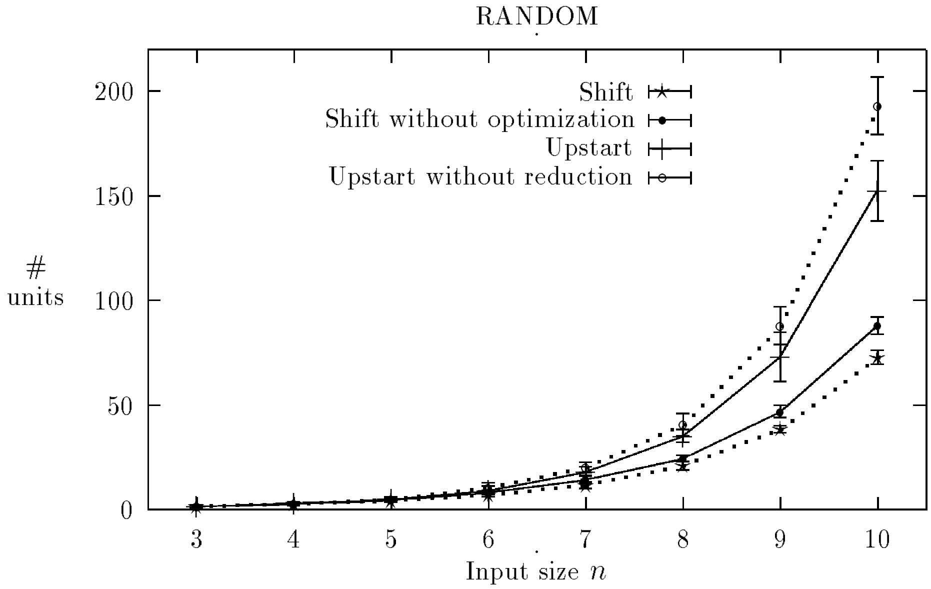


Figure 6.9: Comparison between the SHIFT and UPSTART variants. Average number of units over 25 different random tasks. UPSTART without task reduction allows to contract the tree-like topology in a single hidden layer.

without any particular structure.

For all methods, the computational cost is roughly proportional to the number of units. This is a fairly good estimate for SHIFT and UPSTART in which all units have the same number of inputs (namely n). However, the actual computational requirements may be much larger for PTI or TILING, where the units allocated in the successive layers may have a number of inputs much larger than n.

6.6.2 Generalization

The generalization properties of the constructed networks have been tested on two structured problems. The first one is the classical *K-or-more clumps* problem that is frequently used as a benchmark [52]. Considering binary input vectors $\mathbf{a}^k \in \{0,1\}^n$ as strings of 0 and 1, this predicate associates an output 1 to every input vector that has K or more contiguous blocks of 1 and 0 otherwise. A cyclical boundary condition is used that makes the first and last components adjacent. Although the case $K = 2$ has sometimes been considered [61], we take $K = 3$. Indeed, for $K = 2$ there are only $n(n-1)+1$ possible input vectors with at most 1 clump. If $n = 25$, like in [61], considering tasks with $p = 600$

examples of which half of them are negative amounts to select half of all possible negative examples. In such a case, every system which just stores the negative examples and provides a positive answer for all the others, is guaranteed to make at most 25% of errors on any training set of the same size. For $K = 3$, the class of negative examples is much larger and hence the generalization problem harder.

The results obtained for the 3-or-more clumps problem with $n = 25$ are reported in Figures 6.10 and 6.11 as well as in Table 6.1. The tasks were generated through a stochastic procedure [2] [38] ensuring that the average number of clumps is equal to 2.5. Thus approximately half of the examples are positive and half are negative. In order to evaluate the generalization abilities in terms of the available information, tasks of various sizes are considered. Generalization rates are estimated by measuring the percentage of misclassified vectors on a testing set, which is of the same size p as the task and is generated using the same procedure.

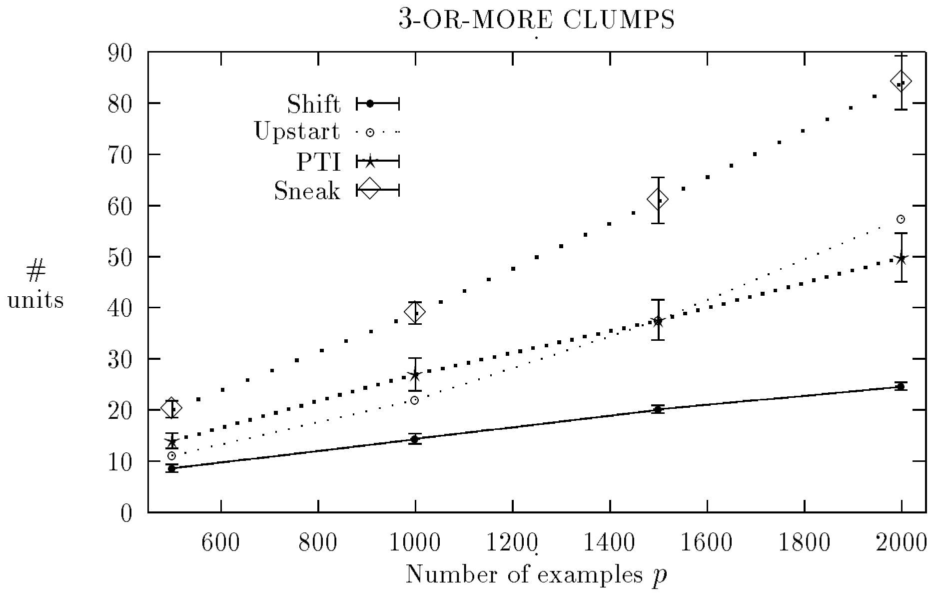


Figure 6.10: 3-or-more clumps tasks with $n = 25$ inputs and various number of examples p. Average number of units in the networks constructed by the different algorithms. Averages are over 25 different tasks.

The SHIFT algorithm produces networks that are much smaller than those obtained with all the other algorithms and that exhibit better generalization abilities. In spite of

[2] The code of the Monte Carlo procedure was kindly provided by Marcus Frean.

the difficulty of the problem, an average percentage of error of nearly 25% is reached for tasks with $p = 2000$ examples. Moreover, the networks have a single hidden layer.

For clarity, the standard deviations relative to UPSTART are not plotted in Figure 6.10. As shown in Table 6.1, they are very large. This fact points out a further disadvantage of this algorithm. The size of the network constructed by UPSTART can vary enormously for tasks of the same type and the same sizes. The situation is different for our three algorithms. In particular, for SHIFT the standard deviations are very small.

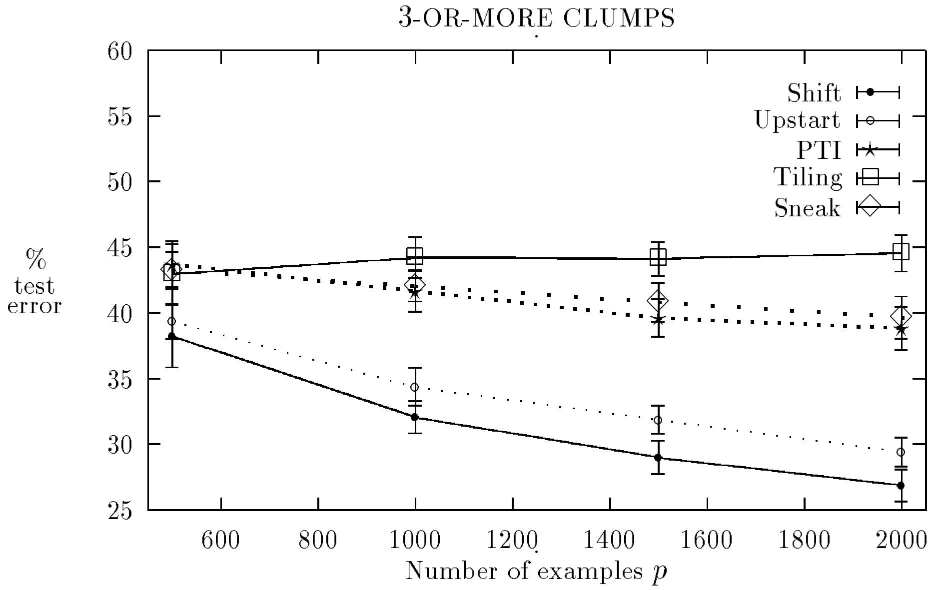


Figure 6.11: Generalization for 3-or-more clumps tasks with $n = 25$ inputs. Average percetange of errors on a testing set having the same size p as the training set. Averages are over 25 different tasks.

The size of the networks produced by the TILING algorithm is not reported in Figure 6.10 because it is very large. As indicated in Table 6.1, it is greater than 550 for tasks with $p = 2000$ examples. It is worth noting that, unlike in all other algorithms, the size of the TILING networks grows so fast with respect to the number of examples p that the average percentage of test error increases with p.

According to Table 6.1, the SHIFT algorithm compares very favorably to UPSTART when the version without task reduction that leads to the same type of two-layer networks is considered. The size of the networks is much larger and they have considerably worse generalization abilities.

algorithm	units	weights	layers	% test error
Shift	25.6 ± 1.0	664.6 ± 27.5	1	26.2 ± 1.1
Upstart	57.4 ± 33.5	1492.4 ± 873.1	22.1 ± 23.1	29.4 ± 1.1
Upstart wr	94.0 ± 70.5	2444.0 ± 1832.6	34.8 ± 30.4	32.8 ± 1.6
PTI	49.8 ± 4.8	1276.0 ± 170.7	4.3 ± 0.5	38.8 ± 1.6
Tiling	559.6 ± 34.4	11322.0 ± 685.1	30.4 ± 2.0	44.6 ± 1.4
Sneak	84.0 ± 5.3	2806.0 ± 288.8	1	39.7 ± 1.6

Table 6.1: 3-or-more clumps tasks with $n = 25$ inputs and $p = 2000$ examples. Size of the networks in terms of units, weights and layers as well as the percentage of errors on a testing set having the same size p as the training set. Averages are over 25 different tasks. For the Sneak algorithm, the number of weights includes those needed to compute the parity function. Upstart wr corresponds to the Upstart variant without task reduction.

As for random tasks, the PTI algorithm performs much better than the Tiling. The constructed networks are more compact than those obtained with all other methods except Shift, but they have poor generalization properties. The comparison between PTI and Upstart is interesting. Although the resulting networks are of comparable size, their performances on the testing sets differ substantially. This may indicate that the networks built by PTI tend to have richer representation capabilities than the tree-like ones generated by Upstart.

Finally, the Sneak algorithm performs slightly worse than PTI in terms of network size and of generalization.

As second test problem, we propose an extension of the well-known symmetry problem that we refer to as *K-similarity*. The objective is to classify binary vectors of even size n depending on whether or not the two subvectors defined by the first $n/2$ components and the last $n/2$ ones have a Hamming distance at most K. Clearly, 0-similarity corresponds to the usual symmetry problem.

The results obtained for 5-similarity tasks with up to $p = 3000$ examples are reported in Figures 6.12 and 6.13 and in Table 6.2. Notice that in this case the class of positive examples (i.e with Hamming distance at most K) is of cardinality

$$2^{n/2} \sum_{i=0}^{K} \binom{n/2}{i} .$$

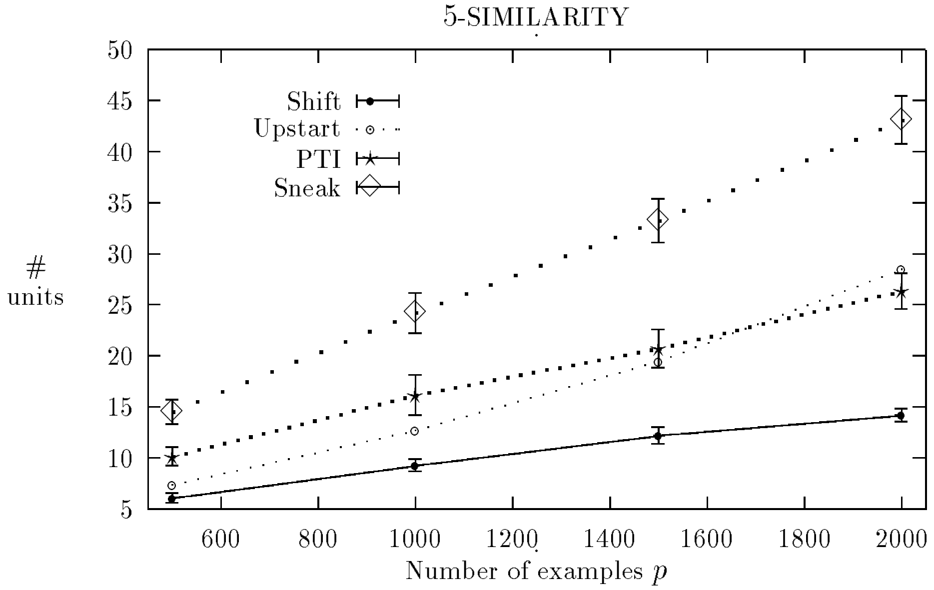


Figure 6.12: 5-similarity with $n = 30$ and various number of examples p. Average number of units in the networks constructed by the different algorithms. Averages are over 25 different tasks.

Since this number is of the order of 10^8 for $n = 30$, tasks with up to $p = 3000$ examples contain only a very small percentage of examples of the smallest class. Thus, error rates on the testing sets are meaningful.

The overall results are very similar to those obtained for the 3-or-more clumps problems. The relative differences between the various algorithms are confirmed. The SHIFT strategy performs best in terms of size of the networks as well as of their generalization performances. For $p = 2000$, the average percentage of testing error is smaller than 20%. Also in this case, the PTI algorithm compares very favorably to the TILING.

All previous results have been obtained by training each unit with 2000 cycles of Frean TPA version. A good initial temperature has been selected for each task using a trial-and-error procedure. Depending on the problem, it ranges from 1 to 4. If single units are trained with an adaptive TPA versions, at least as good results are obtained. Moreover, the global construction process is faster since a smaller number of cycles (e.g. 500) suffices.

Figure 6.14 shows the typical impact of an efficient TPA version on the size of the networks produced by the SHIFT algorithm. The 3-or-more clumps problem is considered and the local training procedures are applied for 500 cycles. For Frean TPA version with fixed initial temperature, a good value $t_0 = 3.5$ has been chosen.

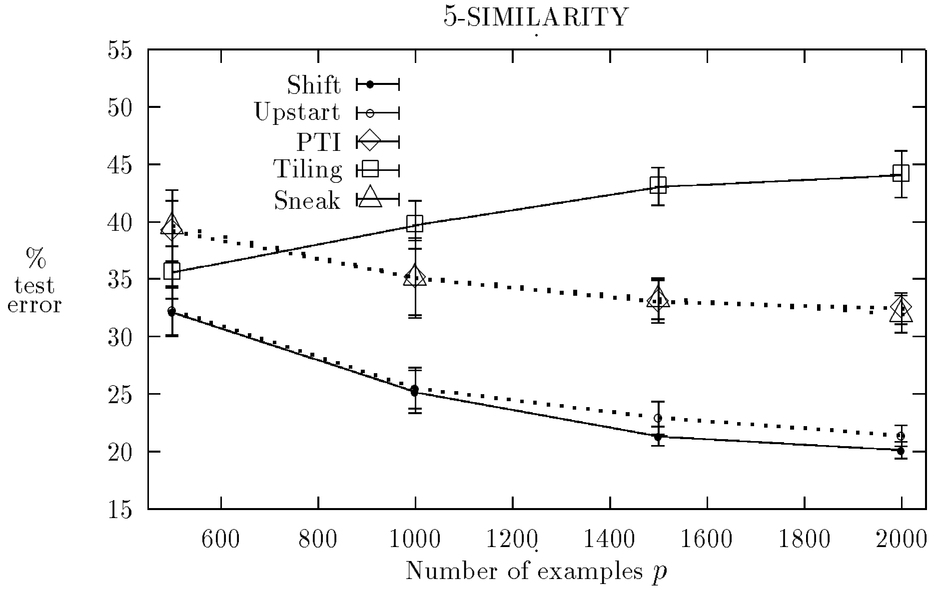


Figure 6.13: Generalization for 5-similarity tasks with $n = 25$ inputs and various number of examples p. Averages percetange of errors on a testing set having the same size p as the training set. Averages are over 25 different tasks.

algorithm	units	weights	layers	% test error
Shift	14.7 ± 0.8	456.6 ± 23.7	1	19.6 ± 0.9
Upstart	28.4 ± 11.7	880.6 ± 363.0	11.5 ± 8.2	21.4 ± 0.9
Upstart wr	56.3 ± 42.4	1743.7 ± 1313.6	26.7 ± 27.8	25.6 ± 1.0
PTI	26.4 ± 1.8	694.9 ± 35.9	3.6 ± 0.5	32.5 ± 1.4
Tiling	544.9 ± 46.1	11117.3 ± 997.4	29.6 ± 2.2	44.1 ± 2.0
Sneak	43.1 ± 2.4	1096.3 ± 88.4	1	32.0 ± 1.6

Table 6.2: 5-similarity tasks with $n = 30$ inputs and $p = 2000$ examples. Size of the networks in terms of units, weights and layers as well as the percentage of errors on a testing set having the same size p as the training set. Averages are over 25 different tasks. For the Sneak algorithm, the number of weights includes those needed to compute the parity function. Upstart wr corresponds to the Upstart variant without task reduction.

Finally, note that the computation time needed to build the various networks with the different algorithms is roughly proportional to the size of the network. However,

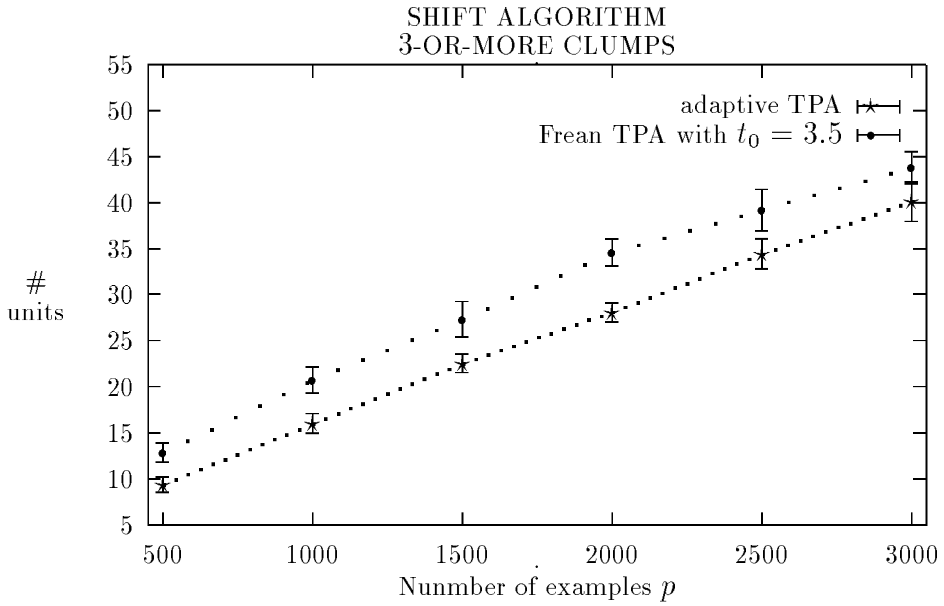


Figure 6.14: Size of the networks produces by the SHIFT algorithm for the 3-or-more clumps tasks with $n = 25$ inputs depending on the TPA variant used for training single units

when the number of allocated units varies considerably, like in TILING, the computational requirements may be much higher.

6.6.3 Real-world problem

In order to test the performance of SHIFT on real-world problems with real-valued inputs, we consider the medical diagnosis application studied in [166]. The data set [3] is relative to breast cancer; it consists of 699 examples from two different classes referring to malignant and benign cases. The nine attributes correspond to nine cellular measurements made microscopically by a surgical oncologist. Measurements are coded as integer values between 1 and 10. The purpose is to separate benign cases from malignant ones.

This data set is a basis of interesting comparison since it has been used to demonstrate the applicability of the Multisurface method [121]. This Linear Programming-trained neural network is currently in use at University of Wisconsin Hospitals.

Before turning to the results, we must mention how SHIFT can be adapted to deal with real-valued inputs. As seen in the previous sections, the guarantee of convergence of

[3] This breast cancer database, provided by Dr. William H. Wolberg, University of Wisconsin Hospitals, Madison, is available at the University of California Irvine repository of Machine Learning.

constructive methods relies on the condition that it is possible to separate any given input vector from all the others. Although such a trivial solution always exists for binary input vectors, this is not necessarily true when real-valued inputs are considered. However, the problem can be overcome by using the stereographic projection as a preprocessing technique [148]. Any given point on a hypersphere being linearly separable from any set of other points, the above condition is met at the simple cost of adding an input dimension.

In [148] the stereographic preprocessing has been shown to work well for the famous benchmark problem that consists in separating two interlocking spirals [58]. This type of task has a circle-like structure and is well-suited to such a transformation. The question arises as to whether, using such a preprocessing, SHIFT can perform well on real-world data with real-valued inputs like in the breast cancer database.

If an adaptive TPA method is used to train single units and only 300 examples of the database are considered, SHIFT builds a network with only 5 hidden units. The same size allows to correctly classify the first 500 examples. If the whole data set is considered, 6 units suffice. These results were obtained by applying the TPA for only 1000 cycles. For 500 cycles, the resulting networks contain at most 2 additional units. In all our experiments the number of hidden units was never greater than 7. For the sake of comparison, the Multisurface method trained on the first 369 examples generated a network with 7 hidden units [122]. The networks built by SHIFT are thus more compact in terms of units. Nevertheless, they have an additional input unit (due to the stereographic preprocessing) and direct input-to-output connections.

In this kind of application, the network ability to correctly classify new cases is of course a major concern. In [122] the authors report that the network trained with the Multisurface method on the first 369 examples correctly classified the following 45 cases. Since the data set has been slightly modified, we have carried out other tests in order to evaluate the generalization properties of the networks produced by the SHIFT algorithm. In particular, we used the first 400, 500 and respectively 654 examples for training and the other ones for testing. In all cases, including the first one where $p = 400$ and hence the training set contains less than 2/3 of the examples, the resulting networks made zero error on the unseen examples.

It is worth noting that these results have been obtained with the simplest version of the SHIFT algorithm without any optimization procedure. This indicates that SHIFT can also be successfully applied to real-world problems with real-valued inputs.

6.7 Concluding remarks

The experiments we have carried out for tasks with binary and real inputs show that, in spite of their limitations, greedy strategies which use efficient TPA procedures for training single units can design compact networks with good generalization abilities. Both the Shift and PTI algorithms compare favorably with competing methods for building multilayer networks that derive from similar types of strategies.

Since constructive algorithms are based on a greedy approach, it is not necessary and even desirable to train each unit optimally. Using simple TPA variants, our Shift algorithm constructs rapidly compact networks with a single hidden layer that exhibit good performances on unseen data. This suggests that sophisticated techniques like Linear Programming are not required within simple greedy strategies. This is good news from the hardware implementation point of view.

In spite of the poor generalization performances of PTI networks, algorithms that generate an arbitrary number of hidden layers are worth pursuing because some functions may have a much more compact representation using a larger number of hidden layers. For threshold units with polynomially bounded weights, there exists even a function that can be computed by a three-layer network with a polynomial number of units but that requires an exponential number of units if only a single hidden layer is available [80]. The question of whether similar functions do also exist for unbounded weights is an important open problem in circuit complexity.

Finally, the algorithms we have presented can also be extended to construct networks with multiple outputs. Although the current versions can be applied to each output unit separately, exploiting correlations would certainly lead to more compact networks with better generalization abilities. In Shift framework, the tasks assigned to the new hidden units should aim at correcting the errors of several output units simultaneously.

Conclusion

Finding solutions of linear systems, which satisfy a maximum number of relations or violate a minimum number of them, is a fundamental problem that is attracting a considerable interest in a wide range of disciplines and fields of application. Depending on the particular framework, one is faced with a number of weighted or constrained versions of these two complementary combinatorial problems.

Considering all these variants within a unified framework, we have thoroughly investigated the inherent complexity of the two classes of problems Max FLS and Min ULR, independently from any particular algorithm. Since approximate solutions are of great practical value, we have focused on their degree of approximability. This has been achieved by building on recent advances in computational complexity. The collection of our results provides important insights into the nature of these problems as well as some useful guidelines for devising approximation algorithms. At present, further research should aim at bridging the gap between our upper and lower approximability bounds.

The particular cases of Max FLS and Min ULR arising in the field of artificial neural networks are closely related to the central problem of learning in feedforward networks. Our complexity results bring a fresh look on the hardness of training and designing single-layer networks. Moreover, when considered in the broader framework of computational learning theory, they lead to an extension of the famous PAC model of learning from examples. By proposing a systematic discrimination between the two possible measures of performance, we have made a necessary step towards a more realistic model of approximate learning.

The proven complexity of training even single-layer networks and designing compact perceptrons, compels us to look for efficient heuristics. Since the development of neural network applications depends on the availability of specific hardware, we have focused on simple methods that require local information and are thus better candidates for actual implementations.

The new probabilistic variants of the thermal perceptron methods we have proposed fit into this perspective. While minimizing the number of perceptron errors similarly to TPA, they require lower precision computations and have a guarantee of asymptotic convergence. Although the thermal and probabilistic approaches have here been considered for single perceptrons, they can be extended to multilayer networks. This is particularly attractive for classification problems where the real objective is to minimize the number of misclassifications.

In order to avoid the pitfalls of conventional training algorithms, constructive strategies can be adopted to design multilayer networks. Since all of them are based on a greedy principle, it is not necessary and even desirable to train each unit either optimally or by using sophisticated techniques. If an efficient TPA variant is used, our SHIFT algorithm produces rapidly compact networks with a single hidden layer that have good generalization properties. Experimental results show that SHIFT compares favorably with various competing methods for constructing multilayer networks. Moreover, it can deal with real-valued inputs and can be extended to problems with multiple outputs. Given its simplicity and efficiency, an hardware implementation should be considered.

Appendix: List of problems

Decision problems

SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses, does there exist a truth assignment for X satisfying all the clauses of $\mathcal{C}$?

EXACT 3-SETS COVER: Given a set S with $|S| = 3q$ elements and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets $C_j \subseteq S$ with $|C_j| = 3$ for $1 \leq j \leq m$, does $\mathcal{C}$ contain an exact cover, i.e. $\mathcal{C}' \subseteq \mathcal{C}$ such that each element s_i of S belongs to exactly one element of $\mathcal{C}'$?

Optimization problems

Computational learning theory

MIN $\mathcal{H}$-DISAGREEMENT: Given a parameterized hypothesis space $\mathcal{H} = \cup_{n \geq 1} \mathcal{H}_n$ and a set of p examples of size n, find a hypothesis $h \in \mathcal{H}_n$ which misclassifies as few examples as possible.

MAX $\mathcal{H}$-CONSISTENCY: Given a parameterized hypothesis space $\mathcal{H} = \cup_{n \geq 1} \mathcal{H}_n$ and a set of p examples of size n, find a hypothesis $h \in \mathcal{H}_n$ which correctly classifies as many examples as possible.

MIN $\mathcal{P}$-DISAGREEMENT: Given a task $T \subseteq \mathbb{R}^n \times \mathbb{B}$, find a perceptron $(\mathbf{w}, w_0) \in \mathbb{R}^{n+1}$ which misclassifies as few examples as possible.

MAX $\mathcal{P}$-CONSISTENCY question: Given a task $T \subseteq \mathbb{R}^n \times \mathbb{B}$, find a perceptron $(\mathbf{w}, w_0) \in \mathbb{R}^{n+1}$ which is consistent with as many examples as possible.

Optimum Perceptron: Given a task $T \subseteq \mathbb{R}^n \times \mathbb{B}$, find a perceptron $(\mathbf{w}, w_0) \in \mathbb{R}^{n+1}$ which is consistent with T and has as few nonzero weights w_j, $1 \leq j \leq n$, as possible.

Graph theory

Max Cut: Given an undirected graph $G = (V, E)$, find a set of nodes $V' \subseteq V$ such that the number of edges $[v_i, v_j] \in E$ with $v_i \in V'$ and $v_j \in V - V'$ is as large as possible.

Min Node Cover: Given an undirected graph $G = (V, E)$, find a set of nodes $V' \subseteq V$ of minimum cardinality which is incident to every edge, i.e. for each $[v_1, v_2] \in E$ at least one extremity is in V'.

Min Dominating Set: Given an undirected graph $G = (V, E)$, find a dominating set $V' \subseteq V$ of minimum cardinality, i.e. for all $v \in V - V'$ there exists $v' \in V'$ such that $[v, v'] \in E$.

Max Ind Set: Given an undirected graph $G = (V, E)$, find a maximum independent set of nodes, i.e. a set of nodes that are not pairwise adjacent.

Min Ind Dom Set: Given an undirected graph $G = (V, E)$, find an independent set V' of minimum cardinality which dominates all nodes of G.

Min Graph Colouring: Given an undirected graph $G = (V, E)$, color the nodes of G with as few colors as possible so that no two adjacent nodes receive the same color.

Longest Path: Given an undirected graph $G = (V, E)$, find a longest simple path in G.

Min Tsp: Given a directed graph $G = (V, U)$ with a nonnegative distance d_{ij} attached to each arc (v_i, v_j) going from v_i to v_j, find a shortest tour which visits every node exactly once.

Logic

MAX SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

MAX 2SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most 2 literals in each clause, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

MAX 3SAT: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most three literals, find a truth assignment for X which satisfies as many clauses of $\mathcal{C}$ as possible.

MAX ONES: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most 3 literals, find a truth assignment for X that satisfies every clause and that contains as many TRUE variables as possible.

MAX 2 ONES NEG: Given a finite set X of variables and a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of 2-literal clauses with only negated variables, find a truth assignment for X which satisfies every clause and which contains as many TRUE variables as possible.

MAX DONES: Given two finite sets X, Z of variables and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of disjunctive clauses with at most 3 literals, find a truth assignment for X and Z that satisfies every clause and contains as many Z variables set to TRUE as possible.

Mathematical programming

MAX FLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A and an n-dimensional vector $\mathbf{b}$ both with rational coefficients, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which satisfies as many relations as possible.

MIN ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A and an n-dimensional vector $\mathbf{b}$ both with rational coefficients, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which violates as few relations as possible while satisfying all the others.

Open Hemisphere: Given a finite set $\{\mathbf{a}^k\}_{1\leq k\leq p}$ of n-dimensional vectors with rational components, find a vector $\mathbf{x} \in \mathbb{Q}^n$ such that $\mathbf{a}^k\mathbf{x} > 0$ for as many $\mathbf{a}^k$ as possible.

Closed Hemisphere: Given a finite set $\{\mathbf{a}^k\}_{1\leq k\leq p}$ of n-dimensional vectors with rational components, find a vector $\mathbf{x} \in \mathbb{Q}^n$ with $\mathbf{x} \neq \mathbf{0}$ such that $\mathbf{a}^k\mathbf{x} \geq 0$ for as many $\mathbf{a}^k$ as possible.

Threshold Min ULR$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system as in Min ULR$^{\mathcal{R}}$ and a lower bound T on the minimum number of unsatisfiable relations, find a solution $\mathbf{x} \in \mathbb{Q}^n$ that minimizes the difference between the number s of relations it violates and the bound T, that is $s - T$.

C Max FLS with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p\times n$ matrix A and a subset of mandatory relations, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which satisfies all mandatory relations and as many optional relations as possible.

C Min ULR with $\mathcal{R} \in \{=, \geq, >, \neq\}$: Given a linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a $p \times n$ matrix A and a subset of mandatory relations, find a solution $\mathbf{x} \in \mathbb{Q}^n$ which satisfies all mandatory relations and violates as few optional relations as possible while satisfying all the others.

Min RVLS$^{\mathcal{R}}$ with $\mathcal{R} \in \{=, \geq, >\}$: Given a feasible linear system $A\mathbf{x}\,\mathcal{R}\,\mathbf{b}$ with a matrix A of size $p \times n$, find a solution $\mathbf{x} \in \mathbb{Q}^n$ satisfying all relations with as few nonzero variables as possible.

Max Knapsack: Given a finite set U, for each $u \in U$ a size $s(u) \in \mathbb{Z}^+$ as well as a value $v(u) \in \mathbb{Z}^+$ and a positive integer B, find a subset $U' \subseteq U$ such that $\sum_{u\in U'} s(u) \leq B$ and $\sum_{u\in U'} v(u)$ is maximum.

Max PB $0-1$ Programming: Given an $m \times n$ matrix A and an m-dimensional vector $\mathbf{b}$, both with ternary components in $\{-1, 0, 1\}$, find an n-dimensional binary vector $\mathbf{x}$ which satisfies $A\mathbf{x} \leq \mathbf{b}$ and has as many nonzero components as possible.

Min PB $0-1$ Programming: Given an $m \times n$ matrix A and an m-dimensional vector $\mathbf{b}$, both with ternary components in $\{-1, 0, 1\}$, find an n-dimensional binary vector $\mathbf{x}$ which satisfies $A\mathbf{x} \geq \mathbf{b}$ and has as few nonzero components as possible.

Sets and partitions

SET COVERING: Given a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of subsets of a finite set S, find a minimum *cover* of S, i.e. a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ of minimum cardinality such that $\cup_{i=1}^{m} C_{j_i} = S$ with $m \leq n$. A cover $\mathcal{C}'$ with all pairwise disjoint sets is an *exact cover*.

Bibliography

[1] N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. In *Proc. of 3rd Ann. Workshop on Computational Learning Theory*, pages 52–66, 1990.

[2] Y. S. Abu-Mostafa and J. M. St.Jacques. Information capacity of the Hopfield model. *IEEE Transactions on Information Theory*, 31:461–464, 1985.

[3] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392, 1954.

[4] N. Alon, 1993. Unpublished note.

[5] N. Alon and F. R. K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72:15–19, 1988.

[6] E. Amaldi. On the complexity of training perceptrons. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 55–60, Amsterdam, 1991. Elsevier science publishers B.V.

[7] E. Amaldi. Complexity of problems related to training perceptrons. In M. Chaleyat-Maurel, M. Cottrell, and J.-C. Fort, editors, *Actes du congrès "Aspects théoriques des réseaux de neurones", Congrès Européen de Mathématiques.* Paris, July 1992.

[8] E. Amaldi and C. G. Diderich. On the probabilistic and thermal perceptron training algorithms. Technical Report ORWP-8-94, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1994.

[9] E. Amaldi and B. Guenin. Constructive methods for designing compact feedforward networks of threshold units. Manuscript in preparation, 1994.

[10] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. Technical Report ORWP-11-93, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne and Technical Report TRITA-NA-9313, ISSN 0348-2952, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1993. To appear in Theoretical Computer Science.

[11] E. Amaldi and V. Kann. On the approximability of finding maximum feasible subsystems of linear systems. In *Proc. 11th Ann. Symp. on Theoretical Aspects of Comput. Sci.*, pages 521–532. Springer-Verlag, 1994. Lecture Notes in Comput. Sci. 775.

[12] E. Amaldi and V. Kann. On the approximability of removing the smallest number of relations from linear systems to achieve feasibility. Technical Report ORWP-6-94, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1994.

[13] E. Amaldi, E. Mayoraz, and D. de Werra. A review of combinatorial problems arising in feedforward neural network design. *Discrete Applied Mathematics*, 52:111–138, 1994.

[14] E. Amaldi and S. Nicolis. Stability-capacity diagram of a neural network with ising bonds. *Journal of Physics (France)*, 50:2333–2345, 1989.

[15] D. Amit. *Modeling brain function: the world of attractor neural networks.* Cambridge University Press, 1989.

[16] D. Angluin and C. H. Smith. Inductive inference: theory and methods. *ACM Computing Surveys*, 15:237–269, 1983.

[17] M. Anthony and N. Biggs. *Computational learning theory: An introduction.* Cambridge Tracts in Theoretical Computer Science 30, Cambridge University Press, 1992.

[18] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Proc. of 34rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, pages 724–733, 1993.

[19] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. Manuscript, 1994.

[20] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. of 33rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, pages 14–23, 1992.

[21] G. Ausiello, P. Crescenzi, and M. Protasi. Approximate solutions of NP optimization problems. Technical Report SI/RR-94/03, Università di Roma "La Sapienza", 1994.

[22] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi. Toward a unified approach for the classification of NP-complete optimization problems. *Theoretical Computer Science*, 12:83–96, 1980.

[23] G. Ausiello and M. Protasi. NP optimization problems and local optima. Technical report, Università di Roma "La Sapienza" (ESPRIT Alcom II report), 1993.

[24] L. Babai. Transparent proofs and limits to approximation. In *Proceedings of the International Congress of Mathematics (ICM'94)*. Birhäuser Verlag, to appear in 1995.

[25] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. In *Proc. of 24th Ann. IEEE Symp. on Foundations of Comput. Sci.*, pages 265–273, 1983.

[26] R. Bar-Yehuda and S. Even. *A local-ratio theorem for approximating the weighted vertex cover problem*, volume 25 of *Annals of Discrete Mathematics*, pages 27–46. Elsevier science publishers B.V., Amsterdam, 1985.

[27] E. Barkai, D. Hansel, and I. Kanter. Statistical mechanics of a multilayer neural networks. *Physics Review Letters*, 65:2312–2315, 1990.

[28] E. Barkai and I. Kanter. Storage capacity of a multilayer neural network with binary weights. *Europhysics Letters*, 14:107–112, 1991.

[29] P. L. Bartlett. Vapnik-Chervonenkis dimension bounds for two- and three-layer networks. *Neural Computation*, 5:371–373, 1993.

[30] E. B. Baum. A proposal for more powerful learning algorithms. *Neural Computation*, 1:201–207, 1989.

[31] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[32] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proc. Twenty fifth Ann. ACM Symp. on Theory of Comp.*, pages 294–304. ACM, 1993.

[33] M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. Twenty sixth Ann. ACM Symp. on Theory of Comp.*, pages 184–193, 1994.

[34] A. Ben-Israel and T. N. E. Greville. *Generalized inverses: theory and applications.* Wiley and Sons, New York, 1974.

[35] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[36] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Proc. 6th Ann. Symp. on Theoretical Aspects of Comput. Sci.*, pages 256–268. Springer-Verlag, 1989. Lecture Notes in Comput. Sci. 349.

[37] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992.

[38] K. Binder. *Monte Carlo methods in statistical Physics.* Springer-Verlag, Berlin, 1979. Topics in Current Physics 7.

[39] H. D. Block and S. A. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. In *Proceedings of American Mathematical Society*, pages 229–235, 1970.

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

[41] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of ACM*, 36:929–965, 1989.

[42] N. Burgess, M. N. Granieri, and S. Patarnello. 3-D object classification: Application of a constructive algorithm. *International Journal of Neural Systems*, 2:275–282, 1991.

[43] J. Chinneck and E. Dravnieks. Locating minimal infeasible constraint sets in linear programs. *ORSA Journal on Computing*, 3:157–168, 1991.

[44] V. Chvatal. *Linear programming.* Freeman and Co., New York, 1983.

[45] D. Cohn and G. Tesauro. How tight are the Vapnik-Chervonenkis bounds? *Neural Computation*, 4:249–269, 1992.

[46] S. A. Cook. The complexity of theorem proving procedures. In *Proc. Third Ann. ACM Symp. on Theory of Comp*, pages 151–158. ACM, 1971.

[47] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Manuscript, 1994.

[48] P. Crescenzi, V. Kann, and L. Trevisan. On the structure of the classes NPO and APX. Manuscript in preparation, 1994.

[49] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93(2):241–262, 1991.

[50] Y. Le Cum, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[51] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2:303–314, 1989.

[52] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield. Large automatic learning, rule extraction, and generalization. *Complex Systems*, 1:877–922, 1987.

[53] C. G. Diderich. Détermination de sous-systèmes linéaires compatibles de taille maximale. Diploma Project, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1993.

[54] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. Wiley and Sons, New York, 1973.

[55] J. Edmonds. Redundency and Helly for linear programming. Lecture notes, April 1994.

[56] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–261, 1989.

[57] R. Fagin. Generalized first-order spectra, and polynomial-time recognizable sets. In R. Karp, editor, *Complexity and Computations*. AMS, 1974.

[58] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University, 1990.

[59] U. Feige and L. Lovász. Two–prover one round proof systems: Their power and their problems. In *Proc. Twenty fifth Ann. ACM Symp. on Theory of Comp.*, pages 733–744, 1992.

[60] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[61] M. Frean. *Small Nets and Short Paths*. PhD thesis, Department of Physics, University of Edinburgh, Edinburgh, Scotland, 1990.

[62] M. Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2:198–209, 1990.

[63] M. Frean. A "thermal" perceptron learning rule. *Neural Computation*, 4(6):946–957, 1992.

[64] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research*, 7:44–60, 1981.

[65] S. I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1:179–191, 1990.

[66] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[67] W. V. Gehrlein. General mathemtical programming formulations for statistical classification problems. *Operations Research Letters*, 5:229–304, 1986.

[68] J. Gleeson and J. Ryan. Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing*, 3:61–63, 1990.

[69] F. Glover. Tabu search - part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[70] F. Glover. Improved linear and integer programming models for discriminant analysis. *Decision Sciences*, 21:771–785, 1990.

[71] F. Glover. Tabu search - part II. *ORSA Journal on Computing*, 2:4–32, 1990.

[72] M. X. Goemans and D. P. Williamson. .878–approximation algorithms for MAX CUT and MAX 2SAT. In *Proc. Twenty sixth Ann. ACM Symp. on Theory of Comp.*, pages 422–431, 1994.

[73] D. Goldfarb and M. J. Todd. Linear programming. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Handbooks in OR & MS, Vol.1, Optimization*, pages 73–170, Amsterdam, 1989. Elsevier science publishers B.V.

[74] M. Golea and M. Marchand. A growth algorithm for neural network decision trees. *Europhysics Letters*, 12:205–210, 1990.

[75] H. J. Greenberg. Consistency, redundancy, and implied equalities in linear systems. Manuscript, November 1993.

[76] H. J. Greenberg and F. H. Murphy. Approaches to diagnosing infeasible linear programs. *ORSA Journal on Computing*, 3:253–261, 1991.

[77] R. Greer. *Trees and Hills: Methodology for Maximizing Functions of Systems of Linear Relations*, volume 22 of *Annals of Discrete Mathematics*. Elsevier science publishers B.V., Amsterdam, 1984.

[78] T. Grossman. The CHIR algorithm: A generalization for multiple outputs and multilayer networks. *Complex Systems*, 3:407, 1990.

[79] T. Grossman, R. Meir, and E. Domany. Learning by choice of internal representations. *Complex Systems*, 2:555–574, 1988.

[80] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan. Threshold circuits of bounded depth. In *Proc. of 28th Ann. IEEE Symp. on Foundations of Comput. Sci.*, pages 99–110, 1987.

[81] M. M. Halldórsson. Approximating the minimum maximal independence number. *Inform. Process. Lett.*, 46:169–172, 1993.

[82] D. J. Hand. *Discrimination and classification*. Wiley and Sons, New York, 1981.

[83] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[84] J. Håstad, S. Phillips, and S. Safra. A well-characterized approximation problem. *Inform. Process. Lett.*, 47:301–305, 1993.

[85] J. Hertz, A. Krogh, and G. Palmer. *Introduction to the theory of neural computation.* Addison Wesley, Redwood City CA, 1991.

[86] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[87] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of ACM*, 34(1):144–162, 1987.

[88] K.-U. Höffgen. Computational limitations on training sigmoid neural networks. *Inform. Process. Lett.*, 46:269–274, 1993.

[89] K.-U. Höffgen, H.-U. Simon, and K. van Horn. Robust trainability of single neurons. Technical Report CS-92-9, Computer Science Department, Brigham Young University, Provo, 1992.

[90] K. Hornik, M. Stichcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[91] O. H. Ibarra and C. E. Kim. Fast approximation for the knapsack and sum of subset problems. *Journal of ACM*, 22(4):463–468, 1975.

[92] R. W. Irving. On approximating the minimum independent dominating set. *Inform. Process. Lett.*, 37:197–200, 1991.

[93] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[94] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6:93–107, 1978.

[95] S. J. Judd. Learning in networks is hard. In *Proc. of the First International Conference on Neural Networks*, pages 685–692. IEEE Computer Society Press, 1987.

[96] S. J. Judd. On the complexity of loading shallow neural networks. *Journal of Compexity*, 4:177–192, 1988.

[97] S. J. Judd. *Neural network design and the complexity of learning.* MIT Press, Cambridge, MA, 1990.

[98] N. Kahale. On the second eigenvalue and linear expansion of regular graphs. In *Proc. of 33rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, pages 296–303, 1992.

[99] V. Kann. *On the Approximability of NP-complete Optimization Problems.* PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[100] V. Kann. Polynomially bounded minimization problems which are hard to approximate. In *Proc. of 20th International Colloquium on Automata, Languages and Programming*, pages 52–63. Springer-Verlag, 1993. Lecture Notes in Comput. Sci. 700.

[101] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[102] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[103] R. M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34:165–201, 1991.

[104] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of ACM*, 41:67–95, 1994.

[105] M. J. Kearns, R. E. Schapire, and L. M. Sallie. Toward efficient agnostic learning. In *Proc. of 5th Ann. Workshop on Computational Learning Theory*, pages 341–353, 1992.

[106] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[107] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. Manuscript, December 1993.

[108] G. J. Koehler. Improper linear discriminant classifiers. *European Journal of Operational Research*, 50:188–198, 1991.

[109] G. J. Koehler. Linear discriminant functions determined by genetic search. *ORSA Journal on Computing*, 3:345–357, 1991.

[110] G. J. Koehler and S. S. Erenguc. Minimizing misclassifications in linear discriminant analysis. *Decision Sciences*, 21:63–85, 1990.

[111] T. Kohonen. *Self-organization and associative memory.* Springer series in Information Sciences, 2 edition. Springer-Verlag, 1987.

[112] P. G. Kolaitis and M. N. Thakur. Approximation properties of NP minimization classes. In *Proc. 6th Ann. Conf. on Structures in Computer Science*, pages 353–366, 1991.

[113] P. G. Kolaitis and M. N. Thakur. Logical definability of NP optimization problems. Technical Report UCSC-CRL-93-10, Board of Studies in Computer and Information Sciences, University of California at Santa Cruz, 1993. To be published in Information and Computation.

[114] M. Li and P. M. B. Vitányi. Average case complexity under the universal distribution equals worst–case complexity. *Inform. Process. Lett.*, 42:145–149, 1992.

[115] J.-H. Lin and J. S. Vitter. Complexity results on learning by neural nets. *Machine Learning*, 6:211–230, 1991.

[116] N. Linial, Y. Mansour, and R. L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. *Information and Computation*, 90:33–49, 1991.

[117] R. P. Lippmann. An introduction to computing with neural nets. *IEEE Acoustic Speech and Signal Processing Magazine*, 4:4–22, 1987.

[118] J. N. Van Loon. Irreducibly inconsistent systems. *European Journal of Operational Research*, 8:283–288, 1981.

[119] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[120] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proc. Twenty fifth Ann. ACM Symp. on Theory of Comp.*, pages 286–293, 1993.

[121] O. L. Mangasarian. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5:349–360, 1993.

[122] O. L. Mangasarian, R. Setiono, and W. H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. In T. F. Coleman and Y. Li, editors, *Large-sacle numerical optimization*, pages 22–30. SIAM Publications, 1990.

[123] M. Marchand and M. Golea. On learning simple neural concepts: from halfspace intersections to neural decision lists. *Network: Computation in Neural Systems*, 4:67–85, 1993.

[124] M. Marchand, M. Golea, and P. Ruján. A convergence theorem for sequential learning in two-layer perceptrons. *Europhysics Letters*, 11:487–492, 1990.

[125] D. Martinez and D. Estève. The offset algorithm: Building and learning method for multilayer neural networks. *Europhysics Letters*, 18:95–100, 1992.

[126] E. Mayoraz. *Feedforward boolean neural networks with discrete weights: computational power and training.* PhD thesis, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1993.

[127] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in the nervous system. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[128] G. J. McLachlan. *Discriminant analysis and statistical pattern recognition.* Wiley and Sons, New York, 1992.

[129] M. Mézard and J.-P. Nadal. Learning in feedforward layered networks: the tiling algorithm. *Journal of Physics A: Math. Gen.*, 22:2191–2203, 1989.

[130] M. L. Minsky and S. Papert. *Perceptrons: An introduction to computational Geometry.* MIT Press, Cambridge, MA, 1988. Expanded edition.

[131] G. J. Mitchison and R. M. Durbin. Bounds on the learning capacity of some multilayer networks. *Biological Cybernetics*, 60:345–356, 1989.

[132] R. Motwani. Lecture notes on approximation algorithms. Technical Report No. STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.

[133] T. S. Motzkin and I. J. Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:393–404, 1954.

[134] B. K. Natarajan. *Machine learning: A theoretical approach.* Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[135] P. Orponen and H. Mannila. On approximation preserving reductions: Complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[136] A. Panconesi and D. Ranjan. Quantifiers and approximation. *Theoretical Computer Science*, 107:145–163, 1993.

[137] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice Hall, 1982.

[138] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[139] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 1992.

[140] M. Parker and J. Ryan. Covering IIS's. Manuscript in preparation, 1994.

[141] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science*, 15:251–277, 1981.

[142] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of ACM*, 35:965–984, 1988.

[143] R. Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms.* Spartan Books, New York, 1962.

[144] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel distributed processing, Explorations in the microstructure of cognition*, pages 318–362, Cambridge, MA, 1986. MIT Press.

[145] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[146] D. E. Rumelhart and J. L. McClelland. *Parallel distributed processing, Explorations in the microstructure of cognition.* MIT Press, Cambridge, MA, 1986.

[147] D. Saad and R. Sasson. Examining the CHIR algorithm performance for multilayer networks and continuous input vectors. *International Journal of Neural Systems*, 3:157–165, 1992.

[148] J. Saffery and C. Thorton. Using the stereographic projection as a pre-processing technique for Upstart. In *Proc. International Joint Conference on Neural Networks of the IEEE*, volume II, pages 441–446, 1991.

[149] S. K. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of ACM*, 23(3):555–565, 1976.

[150] M. Schmitt. On the complexity of consistency problems for neurons with binary weights. Technical Report Nr. 94-01, Fakultät für Informatik, Universität Ulm, 1994.

[151] A. Schrijver. *Theory of linear and integer programming.* Interscience series in discrete mathematics and optimization. Wiley and Sons, New York, 1986.

[152] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, pages 145–168, 1987.

[153] R. J. Serfling. *Approximation theorems of mathematical statistics.* Wiley and Sons, New York, 1980.

[154] I. K. Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78:1605–1613, 1990.

[155] J. A. Sirat and J.-P. Nadal. Neural trees: a new tool for classification. *Network: Computation in Neural Systems*, 1:423–438, 1990.

[156] E. D. Sontag. Feedforward nets for interpolation and classification. *Journal of Computer and System Sciences*, 42:20–48, 1992.

[157] A. Stam and E. A. Joachimsthaler. A comparision of a robust mixed–integer approach to existing methods for establishing classification rules for discriminant analysis. *European Journal of Operational Research*, 46:113–122, 1990.

[158] G. Tesauro and R. Janssens. Scaling relationships in backpropagation learning: Dependence on predicate order. *Complex Systems*, 2:39–44, 1988.

[159] L. G. Valiant. A theory of the learnable. *Communications of ACM*, 27:1134–1142, 1984.

[160] K. S. van Horn and T. Martinez. The minimum feature set problem. Technical Report CS-92-8, Computer Science Department, Brigham Young University, Provo, 1992.

[161] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

[162] S. S. Venkatesh. Directed drift: a new linear threshold algorithm for learning binary weights on-line. *Journal of Computer Science and Systems*, 1991.

[163] R. E. Warmack and R. C. Gonzalez. An algorithm for optimal solution of linear inequalities and its application to pattern recognition. *IEEE Transactions on Computers*, 22:1065–1075, 1973.

[164] A. S. Weigend, B. E. Huberman, and D. E. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1:193–209, 1990.

[165] B. Widrow and M. E. Hoff. Adaptative switching circuits. In *1960 IRE WESCON Convention Record*, pages 96–104. IRE, New York, 1960.

[166] W. H. Wolberg and O. L. Mangasarian. Multisurface method for surface separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Science, U.S.A.*, 87:9193–9196, 1990.

[167] M. Yannakakis. On the approximation of maximum satisfiability. In *Proc. Third Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 1–9, 1992.

[168] P. J. Zwietering, E. H. L. Aarts, and J. Wessels. The design and complexity of exact multilayer perceptrons. *International Journal of Neural Systems*, 2:185–199, 1991.

[169] P. J. Zwietering, E. H. L. Aarts, and J. Wessels. Exact classification with two-layered perceptrons. *International Journal of Neural Systems*, 3:143–156, 1992.

Curriculum vitæ

Prénom, nom :	Edoardo Amaldi
Date de naissance :	13 avril 1965
Nationalités :	Italienne et suisse
Etat civil :	Célibataire
Adresse :	Chemin du Coq d'Inde 8A, CH-1223 Cologny

Formation :

1983	Maturité cantonale de type C, Collège Calvin, Genève
1988	Diplôme d'ingénieur mathématicien, Ecole Polytechnique Fédérale de Lausanne (EPFL)
1988	Stage, Università di Roma "La Sapienza", Italie
1988-89	Special graduate student, Computation and Neural Systems Program, California Institute of Technology, Pasadena, USA

Activités professionnelles :

1988-1989	Graduate research assistant, Computation and Neural Systems Program, California Institute of Technology, Pasadena, USA
Dès 1990	Assistant, Chaire de Recherche Opérationnelle, Département de Mathématiques, EPFL

1990-1993	Contribution à la création du Centre MANTRA en systèmes neuro-mimétiques, Département d'Informatique, EPFL
Dès 1991	Chargé de cours dans le cadre de programmes de 3^{e} cycle. Cours postgrade de statistique appliquée (EPFL, 1991). Séminaire du 3^{e} cycle Romand d'Informatique (Charmey, 1991). Cours postgrades en Informatique sur les Réseaux de neurones biologiques et artificiels (EPFL, 1992 et 1993)
1992	Membre du comité scientifique de la conférence “Aspects théoriques des réseaux de neurones” organisée dans le cadre du Congrès Européen de Mathématiques 1992, Paris
1992	Coorganisateur de l'école d'été internationale “Mathematical foundations of artificial neural networks”, Institut Universitaire Kurt Bösch, Sion (septembre)
1992-1993	Chargé de cours en Optimisation (2^{e} cycle), Département de Mathématiques, EPFL
1993	Chercheur invité, School of Operations Research, Cornell University, Ithaca, New York, USA (septembre)
1993-1994	Chargé de cours en Réseaux de neurones artificiels (2^{e} cycle), Département d'Informatique, EPFL
Dès 1994	Assistant, Centre MANTRA en systèmes neuro-mimétiques, Département d'Informatique, EPFL
1994	Chercheur invité, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (juillet).