Theoretical Computer Science 244 (2000) 95–116
Theoretical Computer Science
www.elsevier.com/locate/tcs

Convexity and logical analysis of data

Oya Ekin [a], Peter L. Hammer [b], Alexander Kogan [b,c,*]

[a] *Department of Industrial Engineering, Bilkent University, Ankara, Turkey*
[b] *RUTCOR, Rutgers University, P.O. Box 5062, New Brunswick, NJ 08903-5062, USA*
[c] *Accounting and Information Systems, Faculty of Management, Rutgers University, 180 University Ave., Newark, NJ 07102, USA*

Received January 1998; revised May 1998
Communicated by E. Shamir

Abstract

A Boolean function is called k-convex if for any pair x, y of its true points at Hamming distance at most k, every point "between" x and y is also true. Given a set of true points and a set of false points, the central question of Logical Analysis of Data is the study of those Boolean functions whose values agree with those of the given points. In this paper we examine data sets which admit k-convex Boolean extensions. We provide polynomial algorithms for finding a k-convex extension, if any, and for finding the maximum k for which a k-convex extension exists. We study the problem of uniqueness, and provide a polynomial algorithm for checking whether all k-convex extensions agree in a point outside the given data set. We estimate the number of k-convex Boolean functions, and show that for small k this number is doubly exponential. On the other hand, we also show that for large k the class of k-convex Boolean functions is PAC-learnable. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Partially-defined Boolean functions; Orthogonal disjunctive normal forms; Computational learning theory; Classification; Polynomial algorithms

1. Introduction

Partially defined Boolean functions provide formal representations of data sets arising in numerous applications. Given a set of true points and a set of false points, the central question of logical analysis of data (LAD) is the study of those Boolean functions (called "extensions" of data sets) whose values agree with those of the given points. The basic concepts of LAD are introduced in [9], and an implementation of LAD is described in [6].

A typical data set will usually have exponentially many extensions. In the absence of any additional information about the properties of the data set, the choice of an

* Corresponding author.
E-mail address: kogan@rutcor.rutgers.edu (A. Kogan).

PII: S0304-3975(98)00337-5

extension would be totally arbitrary, and therefore would risk to omit the most significant features of the data set. However, in many practical cases significant information about the data set is available. This information can be used to restrict the set of possible extensions to those satisfying certain required properties. In a typical example, the extension may be required to be a monotone Boolean function, a Boolean function that can be represented as a DNF of low degree (i.e. one consisting only of "short" terms), etc.

It is often known that data points of the same type exhibit certain compactness properties. The property of compactness can be formalized in various ways. For example, if a Boolean function takes the value 1 in two points x and y that are close to each other (e.g. being at Hamming distance at most k), then this function may be required to take the value 1 in every point situated "between" x and y. This property defines the class of so-called k-convex Boolean functions. It turns out that k-convex functions ($k \geqslant 2$) can be characterized by the property that their prime implicants are pairwise strongly orthogonal, i.e. they "conflict" in at least $k+1$ literals. Orthogonal DNFs play an important role in many areas, including operations research (see [12]), reliability theory (see [8, 15]), and computational learning theory (see [3]).

This paper is devoted to the study of data sets which admit k-convex extensions. We provide polynomial algorithms for finding a k-convex extension of a given data set, if any, and for finding the maximum k for which a k-convex extension exists. We also study the problem of uniqueness, and provide a polynomial algorithm for checking whether all k-convex extensions agree in a point which is outside the given data set.

In order to overcome the fact that there are only a very limited number of Boolean functions whose true points and whose false points are both k-convex, we introduce here the concept of k-convex partially-defined Boolean functions, and study the problem of constructing k-convex partially-defined extensions of the given data set.

To study the probabilistic properties of k-convex extensions, we estimate the number of k-convex Boolean functions, and show that for small k this number is doubly exponential. On the other hand, we also show that for large k the class of k-convex Boolean functions is PAC-learnable.

2. Basic concepts

We assume that the reader is familiar with the basic concepts of Boolean algebra, and we only introduce here the notions that we explicitly use in this paper.

2.1. Boolean functions

A *Boolean function* f of n variables $x_1,\dots,x_n$ is a mapping $B^n \to B$, where $B=\{0,1\}$, and where B^n is commonly referred to as the *Boolean hypercube*. The variables $x_1,\dots,x_n$ and their complements $\bar{x}_1, \dots \bar{x}_n$ are called positive and negative *literals*, respectively. We shall sometimes denote x by x^1, and $\bar{x}$ by x^0. For two Boolean functions f and g we write $f \leqslant g$ iff for every 0–1 vector $\boldsymbol{x}$, $f(x_1,\dots,x_n)=1$ implies $g(x_1,\dots,x_n)=1$; in this case g is called a *majorant* of f and f is called a *minorant* of g. Throughout this paper the number of variables will be denoted by n.

The *dual* of a Boolean function $f(\boldsymbol{x})$ is defined as

$$f^{\mathrm{d}}(\boldsymbol{x})=\bar{f}(\bar{\boldsymbol{x}}),$$

where $\bar{\boldsymbol{x}}=(\bar{x}_1,\bar{x}_2,\dots,\bar{x}_n)$ is the *complement* of $\boldsymbol{x}$, and $\bar{f}$ is the complement of f i.e. $\bar{f}(\boldsymbol{y})=1$ if and only if $f(\boldsymbol{y})=0$.

Given a Boolean function, we shall call the points for which $f(\boldsymbol{x})=1(f(\boldsymbol{x}=0)$ the *true points* (*false points*) of the function. The *true* (*false*) *set* of a function f, denoted by $\mathscr{T}_f$ ($\mathscr{F}_f$), is the collection of the true (false) points of f, i.e.

$$\mathscr{T}_f=\{\boldsymbol{x}\in\{0,1\}^n\colon\ f(\boldsymbol{x})=1\} \quad \text{and} \quad \mathscr{F}_f=\{\boldsymbol{x}\in\{0,1\}^n\colon\ f(\boldsymbol{x})=0\}.$$

A *term*, or an *elementary conjunction*, is a conjunction of literals

$$\prod_{i\in P} x_i \prod_{i\in N} \bar{x}_i,$$

where P and N are disjoint subsets of $\{1,\dots,n\}$; by convention, if $P=N=\emptyset$, the term is considered to be the constant 1. The *degree* of a term is $|P|+|N|$. We shall say that a term T *absorbs* another term T', iff $T \vee T'=T$, i.e. iff $T \geqslant T'$ (e.g. the term $x\bar{y}$ absorbs the term $x\bar{y}z$). A term T *covers* a 0–1 point $\boldsymbol{x}^*$ iff $T(\boldsymbol{x}^*)=1$. Given a point $\boldsymbol{s}$, the term $\bigvee_{i=1}^{n} x_i^{s_i}$ is called *minterm*($\boldsymbol{s}$). A term T is called an *implicant* of a function f iff $T \leqslant f$. An implicant T of a function is called *prime* iff there is no distinct implicant T' absorbing T.

A *disjunctive normal form* (DNF) is a disjunction of terms. It is well known that every Boolean function can be represented by a DNF, and that this representation is not unique. A DNF representing a function f is called *prime* iff each term of the DNF is a prime implicant of the function. On the other hand, a DNF representing a function is called *irredundant* iff eliminating any one of its terms results in a DNF which does not represent the same function. Given a DNF Φ, we denote by $|\Phi|$ and *length*(Φ) the number of terms and the number of literals in Φ respectively.

A Boolean function is called *positive* (*negative*) or *monotonically nondecreasing* (*monotonically nonincreasing*) if it has a DNF representation in which each one of the terms consists only of positive (negative) literals.

Two terms are said to be *orthogonal* or to *conflict* in x_i if x_i is a literal in one of them and $\bar{x}_i$ is a literal in the other. If two terms P and Q conflict in exactly one variable, i.e., they have the form $P=x_iP'$ and $Q=\bar{x}_iQ'$ and the elementary conjunctions P' and Q' have no conflict, then the *consensus* of P and Q is defined to be the term $P'Q'$. The *consensus method* applied to an arbitrary DNF Φ performs the following operations as many times as possible:

Consensus: If there exist two terms of Φ having a consensus T which is not absorbed by any term of Φ then replace the DNF Φ by the DNF $\Phi \vee T$.

Absorption: If a term T of Φ absorbs a term T' of Φ, delete T'.

It is easy to notice that all the DNFs produced at every step of the consensus method represent the same function as the original DNF. The following result (see [4, 14]) plays a central role in the theory and applications of Boolean functions:

Proposition 2.1 (Blake [4], Quine [14]). *The consensus method applied to an arbitrary DNF of a Boolean function f results in the DNF which is the disjunction of all the prime implicants of f.*

Throughout the text, the following notation will be used to represent terms:

Definition 2.2. If $S=\{i_1,\dots,i_{|S|}\}\subseteq\{1,\dots,n\}$, and $\boldsymbol{\alpha}_S=(\alpha_{i_1},\dots,\alpha_{i_{|S|}})\in\{0,1\}^{|S|}$ is an "assignment" of 0–1 values to the variables x_i $(i\in S)$, then the term $X^{\boldsymbol{\alpha}_S}$ *associated to* $\boldsymbol{\alpha}_S$ is the conjunction $\Pi_{i\in S}x_i^{\alpha_i}$; if $S=\emptyset$, we define $X^{\emptyset}=1$.

We shall frequently use in this paper the concept of *projection* of a DNF:

Definition 2.3. Let $S=\{i_1,\dots,i_{|S|}\}\subseteq\{1,\dots,n\}$ and let $\boldsymbol{\alpha}=(\alpha_{i_1},\dots,\alpha_{i_{|S|}})\in\{0,1\}^{|S|}$. The *projection* of a DNF $\Phi(x_1,\dots,x_n)$ on $(S,\boldsymbol{\alpha})$ is the DNF $\Phi_{(S,\boldsymbol{\alpha})}$ obtained from Φ by the substitutions $x_i=\alpha_i$ for all $i\in S$.

A classical hard problem concerning Boolean formulae is the *tautology problem* (*TAUT*), which is the Boolean dual of the well known satisfiability problem. The tautology problem can be formulated as follows: given as input a DNF Φ, is there an assignment $\boldsymbol{x}^*\in\{0,1\}^n$ such that $\Phi(\boldsymbol{x}^*)=0$ (i.e. $\boldsymbol{x}^*$ is a *solution* of Φ)?

2.2. Orthogonality

A DNF Φ is called *orthogonal* if every pair of its terms is orthogonal. It is well known that every Boolean function can be represented by an orthogonal DNF (e.g. by its minterm expression). It is also known that there exist Boolean functions that have DNF representations of linear length (in the number of variables), but all of their orthogonal DNFs have exponential length (see [2]). Two DNFs are said to be *orthogonal* if each term of one is orthogonal to all the terms of the other.

Lemma 2.4. *Given two DNFs Φ and Ψ, we can decide in* $O(|\Psi|\,length(\Phi)+|\Phi|\,length(\Psi))$ *time whether they are orthogonal to each other.*

Proof. A comparison of two terms of degree d' and d'' can be done in $O(d'+d'')$ time. Therefore, one can check in $O(length(\Phi)+|\Phi|d_\Psi)$ time whether a term of Ψ having degree d_Ψ is orthogonal to all terms of Φ. Summing this up over all terms of Ψ proves the claimed computational complexity. □

If the DNFs Φ and Ψ depend on n Boolean variables, then $length(\Phi) \leqslant n|\Phi|$ and $length(\Psi) \leqslant n|\Psi|$. Therefore, one can check in $O(n|\Phi||\Psi|)$ time whether Φ and Ψ are orthogonal to each other.

While the result below is perhaps known, we could not find any reference to it.

Proposition 2.5. *Given an orthogonal DNF Φ in n Boolean variables, the TAUT problem for Φ can be solved in $O(length(\Phi))$ time, and a solution $\mathbf{x}$ can be found in $O(length(\Phi)n)$ time. Moreover, one can list all the solutions of Φ in time polynomial in their total number $NTP(\Phi, n)$ and in $length(\Phi)$.*

Proof. We first find the number of true points of Φ. Since no true point is covered by more than one term in Φ, this number is easily computable by simply adding the number of true points covered by each of the terms. Clearly, the answer to the TAUT problem is YES if and only if this number is strictly less than 2^n. The counter needs $n+1$ binary digits, and the number of additions is $|\Phi|$. Since a term of degree d covers exactly 2^{n-d} points, in each addition the counter will be added a binary number whose only 1 appears in position $n-d+1$. The addition of such a number to the counter can be done in $O(d)$ time, since there are only d positions in front of position $n-d+1$. Therefore, the total number of operations is $O(length(\Phi))$.

In order to find a solution of the TAUT problem, if one exists, we find the projections Φ_0 and Φ_1 of Φ on $x_1 = 0$ and on $x_1 = 1$, respectively. Obviously, both Φ_0 and Φ_1 are orthogonal DNFs and at least one of them has a solution (i.e. it does not cover the whole Boolean hypercube B^{n-1}). The recursive application of this procedure to one of the solvable DNFs produced at each step will yield in the end an assignment which solves Φ. Since each step can be done in $O(length(\Phi))$ time, and the number of steps does not exceed the number of variables, the total number of operations is $O(length(\Phi)n)$.

The algorithm described above finds only one assignment which solves Φ. Obviously, if our objective is to obtain all the solutions of the TAUT problem, we shall complete this branching process following not only one but each of the solvable branches. We note that the end result of this process will be a representation of the given Boolean function by a so-called binary decision tree. This algorithm constructs a binary tree in which each path from the root to a 0-leaf represents a partial assignment which solves Φ. $\square$

Note that while one can check in linear time whether an orthogonal DNF is a tautology, by Lemma 2.4 it takes $O(|\Phi| length(\Phi))$ time to check whether the DNF Φ is indeed orthogonal.

2.3. Partially-defined Boolean functions

Definition 2.6. A *partially defined Boolean function* (pdBf) is a pair of disjoint sets (T, F) of Boolean vectors where T denotes a set of true (or positive) points and F denotes a set of false (or negative) points.

Any pdBf (T,F) can be represented by a pair of disjoint DNFs (Φ_T,Φ_F), e.g. by the pair (Φ_T,Φ_F), where

$$\Phi_T = \bigvee_{s\in T}\bigwedge_{i=1}^{n} x_i^{s_i} \quad \text{and} \quad \Phi_F = \bigvee_{s\in F}\bigwedge_{i=1}^{n} x_i^{s_i},$$

i.e. Φ_T and Φ_F are the minterm expansions of the sets T and F respectively.

Example 2.7. Consider the pdBf (T,F), where
$T=\{(11000),(11001),(11010),(11011),(00110)\}$,
$F=\{(01000),(01001),(01100),(01101)\}$.

We can represent this pdBf by the disjoint pair (Φ_T,Φ_F), where

$$\Phi_T = x_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3x_4\bar{x}_5 \quad \text{and} \quad \Phi_F = \bar{x}_1x_2\bar{x}_4.$$

This example shows that the use of DNFs may allow for a more compact representation of pdBfs. However, it will be seen below that the representation of a pdBf by a pair of disjoint sets of points may sometimes allow for polynomial solutions to certain problems which are computationally intractable for the representation by a pair of disjoint DNFs.

Definition 2.8. A *positive pattern* of a pdBf (T,F) is a term which does not cover any points in F and covers at least one point in T. Similarly, a *negative pattern* of a pdBf (T,F) is a term which does not cover any points in T and covers at least one point in F.

Example 2.9. For the pdBf given in Example 2.7, x_1 is a positive pattern and $\bar{x}_1x_2$ is a negative pattern.

Remark 2.10. Given a term S and a pdBf represented by (T,F), it is easy to check whether the term is a positive pattern of the pdBf. We simply verify that the term covers at least one true point and does not cover any of the false points. Similarly, if the pdBf is represented by (Φ_T,Φ_F), it is easy to decide whether a term S is a positive pattern. In this case, S must conflict with every term from Φ_F and there must exist at least one term of Φ_T which does not conflict with S.

Definition 2.11. An *extension* of a pdBf (T,F) is a Boolean function f such that

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } x\in T, \\ 0 & \text{if } x\in F, \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

Definition 2.12. A *positive theory* (or simply, a theory) is an extension which can be represented as a disjunction of positive patterns. Similarly, a *negative theory* is a Boolean function which can be represented as a disjunction of negative patterns.

It is easy to see that a positive theory has a prime DNF representation where each prime implicant is a positive pattern. Note also that a positive theory is not necessarily a positive (i.e. monotonically nondecreasing) function.

Example 2.13. The Boolean functions

$$f = x_1x_2 \vee x_3x_4 \quad \text{and} \quad g = x_1x_2\bar{x}_3 \vee x_3x_4$$

are both positive theories of the pdBf given in Example 2.7. Note that f happens to belong to the class of positive Boolean functions, while g does not.

Remark 2.14. Given a DNF Φ^*, it is easy to check whether it represents a theory for a pdBf (T,F). We simply check that each term of Φ^* is a positive pattern and whether every point in T is covered by some term of Φ^*. However, the same problem becomes hard if the pdBf is represented by (Φ_T, Φ_F), since checking whether every true point of Φ_T is covered by Φ^* is equivalent with solving SAT. In the special case where Φ^* is orthogonal, we can answer this question in polynomial time. We first check (as in Remark 2.10) if each term of Φ^* is a positive pattern. To decide if every true point of Φ_T is covered by Φ^* we have to check whether the inequality

$$\Phi_T \leqslant \Phi^*$$

holds. Equivalently, we have to show that every term T_i of Φ_T satisfies the relation

$$T_i \leqslant \Phi^*.$$

This can be accomplished by substituting in Φ^* the partial assignment corresponding to the term T_i and checking the tautology of the resulting orthogonal DNF (as in Proposition 2.5).

Because of the special role played by various classes of Boolean functions examined in the literature (e.g. monotone, Horn, quadratic, threshold, convex etc.), we shall be frequently interested in extending a pdBf to a fully defined Boolean function belonging to one of these classes.

The central topic of this paper will be the study of the following important problems arising in LAD. Given a pdBf (T,F) and a class of Boolean functions $\mathscr{C}$,

- check whether a theory of (T,F) in $\mathscr{C}$ exists, and if yes, find one;
- checkwhether a theory of (T,F) in $\mathscr{C}$ is unique, and if not, check whether for a given point not belonging to $T \cup F$, all the theories of (T,F) in $\mathscr{C}$ agree.

3. Convex functions

An important property of Boolean functions playing a special role in LAD is that of convexity. Convex Boolean functions were introduced and studied in [11]. In this paper, we shall extend this concept to the case of pdBfs. For the presentation that follows, we shall need several definitions.

The *Hamming distance* between two Boolean vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is the number of components in which they differ:

$$d(\boldsymbol{x},\boldsymbol{y}) = |\{i\colon\, x_i \neq y_i \quad i \in [1,\ldots,n]\}|.$$

Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are called *neighbors* iff $d(\boldsymbol{x},\boldsymbol{y})=1$. A point $\boldsymbol{y}$ is *between* $\boldsymbol{x}$ and $\boldsymbol{z}$ iff $d(\boldsymbol{x},\boldsymbol{y})+d(\boldsymbol{y},\boldsymbol{z})=d(\boldsymbol{x},\boldsymbol{z})$. A sequence of points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k$ is called a *path of length* $k-1$ from $\boldsymbol{x}_1$ to $\boldsymbol{x}_k$ iff any two consecutive points in this sequence are neighbors. A *shortest path* between $\boldsymbol{x}$ and $\boldsymbol{y}$ is a path of length $d(\boldsymbol{x},\boldsymbol{y})$. A *true* (*false*) *path* is a path consisting only of true (false) points of a Boolean function.

We say that two true (false) points are *convexly connected* iff all the shortest paths connecting them are true (false).

For any two terms T and S, let the *distance* between them, denoted by $d(T,S)$, be the number of conflicts between these two terms.

The extremely powerful requirement of convexity puts a severe limitation on the number of functions with this property. In order to provide more flexibility, we introduced in [11] the following relaxation of the definition.

Definition 3.1. For any integer $k \in \{2,\ldots,n\}$, a Boolean function f is called *k-convex* if and only if any pair of true points at distance at most k is convexly connected.

The following results are obtained in [11] and presented here for the sake of completeness.

Proposition 3.2. *For any* $k \geqslant 2$, *a Boolean function* f *is k-convex if and only if any two prime implicants of* f *conflict in at least* $k+1$ *literals.*

Corollary 3.3. *A k-convex Boolean function has a unique prime DNF representation.*

Remark 3.4. A k-convex function having an implicant of degree at most k is equal to that implicant.

With increasing values of k, the statement of Proposition 3.2 gets stronger. In particular, an n-convex function has a single prime implicant. An $(n-1)$-convex function is either an elementary conjunction, or is of the form $x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n} \vee x_1^{\bar{\alpha}_1}x_2^{\bar{\alpha}_2}\ldots x_n^{\bar{\alpha}_n}$.

It follows immediately from the definition of k-convexity that the conjunction of any two k-convex functions is a k-convex function. This justifies the following definition.

Definition 3.5. The *k-convex envelope* of a Boolean function f is the Boolean function $[f]_k$ defined by

(i) $[f]_k$ is k-convex,

(ii) $[f]_k$ is a majorant of f,
(iii) if g is a k-convex majorant of f then $[f]_k$ is a minorant of g.

In other words, the k-convex envelope of f is the smallest k-convex majorant of f.

While the existence and the uniqueness of the k-convex envelope of any Boolean function are obvious, it may be surprising that the DNF representation of the k-convex envelope can be easily constructed from the DNF representation of the original function, as can be seen below.

It is well known that given an arbitrary DNF of a positive Boolean function, we can obtain a positive DNF of it by simply "erasing" all the complemented variables from the given DNF. Obviously, the correctness of this polynomial algorithm is a consequence of the prior knowledge of the positivity of the function.

It is interesting to note that a similarly efficient method can be applied for finding the prime implicants of a Boolean function which is a priori known to be k-convex. We shall use the following:

Definition 3.6. If S and T are elementary conjunctions, then the *convex hull* of S and T is the smallest elementary conjunction $[S,T]$ which satisfies

$$[S,T]\geqslant S \quad \text{and} \quad [S,T]\geqslant T.$$

Specifically, if $S=X^{\alpha_A}X^{\alpha_B}X^{\alpha_C}$ and $T=X^{\bar{\alpha}_A}X^{\alpha_B}X^{\alpha_D}$, then

$$[S,T]=X^{\alpha_B}.$$

Note that when $B=\emptyset$, the convex hull is simply the constant 1. Obviously, $S\vee T\leqslant [S,T]$.

Given any DNF Φ, the *k-convexification method* for finding the k-convex hull of Φ repeats the following step as many times as possible:

- If T_i and T_j are two terms of Φ such that $d(T_i,T_j)\leqslant k$, transform Φ by removing T_i and T_j and adding $[T_i,T_j]$.

The algorithm stops when every two of the remaining terms conflict in at least $k+1$ literals.

The pseudo-code in Fig. 1 provides a careful implementation of the k-convexification method.

It was shown in [11] that the k-convex hull of a DNF represents the k-convex envelope of the function represented by that DNF. Let $[\Phi]_k$ denote the *k-convex hull* of Φ. More precisely, the following results were proven.

Proposition 3.7. *Let a Boolean function f be represented by a DNF Φ. Then the k-convex hull $[\Phi]_k$ is the (unique) irredundant prime DNF of the k-convex envelope $[f]_k$.*

Corollary 3.8. *If f is k-convex for some $k\geqslant 2$ and Φ is an arbitrary DNF of f, then $f=[\Phi]_2$.*

```
Input:           A DNF Φ = ⋁_{i=1}^{m} T_i
Output:          [Φ]_k, the k-convex hull of Φ.
Initialization:  structure terms
                     string term
                     string mode
                     pointer to terms next
                 list of terms List1,List2
                 terms headlist1,current,convhull
                 List1 = [T_1, T_2, ..., T_m]
                 for i = 1 to m
                     List1[i].mode = oldterm
                 List2 = []
Algorithm:       begin {main}
                         while (List1 ⟨ ⟩[]) do
                         headlist1 = List1[1]
                         current = List1[2]
                         while (current ⟨ ⟩∅) do
                             if d(headlist1.term,current.term) ⩽ k
                                 let convhull = [headlist1.term,current.term]
                                 delete headlist1 and current from List1
                                 push convhull to List1
                                 List1[1].mode = newterm
                                 headlist1 = List1[1]
                                 current = List1[2]
                             else current = current.next
                         if headlist1.mode = newterm
                             current = List2[1]
                             while (current ⟨ ⟩∅) do
                                 if d(headlist1.term,current.term) ⩽ k
                                     let convhull = [headlist1,current]
                                     delete headlist1 from List1
                                     delete current from List2
                                     push convhull to List1
                                     List1[1].mode = newterm
                                     headlist1 = List1[1]
                                     current = List2[1]
                                 else current = current.next
                         delete headlist1 from List1
                         push headlist1 to List2
                     end {main}
```

Fig. 1. Procedure k-convexification method (DNF Φ).

Example 3.9. Let

$$\Phi = x_1x_2x_3x_4x_5x_6 \vee x_1x_2x_3x_4x_7 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_5x_8 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4x_6 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_6$$

be the input to the 2-convexification method. Then, the 2-convex hull of Φ is

$$[\Phi]_2 = x_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3.$$

Theorem 3.10. *The k-convex hull $[\Phi]_k$ of an arbitrary DNF Φ can be obtained in $O(n|\Phi|^2)$ time by using the k-convexification method.*

Proof. The k-convexification method described in Fig. 1 maintains two stacks of terms: List1 and List2. All the terms in List1, with the possible exception of the first one, are terms of the original DNF Φ. By construction, every two terms in List2 are at distance at least $k+1$ from each other. After making at most List1 + List2 comparisons of terms, the k-convexification method either

(1) finds a pair of terms at distance at most k, in which case the number of terms in List1 or in List2 is decreased by one, or
(2) moves the first term of List1 to List2.

Since in the beginning the total number of terms in both stacks is $|\Phi|$, the method can make at most $|\Phi|$ steps of type (1). Obviously, after at most $|\Phi|$ steps of type (2) List1 becomes empty, and the method stops. Since List1 + List2 $\leqslant |\Phi|$ and each comparison of two terms can be done in $O(n)$ time, the total running time of the method does not exceed $O(n|\Phi|^2)$. □

4. Convex theories of PDBFs

We shall start now the study of convex theories of partially-defined Boolean functions. The main problems to be analyzed here are those concerning the existence of a k-convex extension, and the determination of the maximum k for which a k-convex extension exists. The other central theme of this section is the recognition of pdBf's having a unique k-convex theory, and, when a k-convex theory is not unique, the recognition of those points where all k-convex theories take the same value.

Theorem 4.1. *A pdBf (Φ_T, Φ_F) has a k-convex extension if and only if the k-convex hull $[\Phi_T]_k$ is orthogonal to Φ_F.*

Proof. Let us consider a pdBf (Φ_T, Φ_F) and a positive integer $k \geqslant 2$. Let us construct the k-convex hull $[\Phi_T]_k$, as described in the previous section. If a k-convex extension ϕ of (Φ_T, Φ_F) exists, then

$$\Phi_T \leqslant [\Phi_T]_k \leqslant \phi,$$

since $[\Phi_T]_k$ is the minimum k-convex majorant of Φ_T.

If $[\Phi_T]_k$ and Φ_F are orthogonal, then $[\Phi_T]_k$ represents a k-convex extension of (Φ_T, Φ_F). If $[\Phi_T]_k$ and Φ_F are not orthogonal, then there must exist $x \in \Phi_F$ such that $[\Phi_T]_k(x) = 1$. In this case, $\phi(x) = 1$, contradicting the assumption that ϕ is an extension. □

Corollary 4.2. *Given a pdBf* (Φ_T, Φ_F) *and a positive integer* $k \geqslant 2$, *we can decide in* $O(n|\Phi_T|(|\Phi_T| + |\Phi_F|)$ *time whether* (Φ_T, Φ_F) *has a k-convex extension, and if so, construct a DNF representing its minimum k-convex theory.*

Proof. Let us define the *k-convex theory algorithm*. It applies the k-convexification method to Φ_T to construct $[\Phi_T]_k$. The algorithm then checks whether $[\Phi_T]_k$ is orthogonal to Φ_F, and if yes, outputs $[\Phi_T]_k$ as the minimum k-convex theory h_k of (Φ_T, Φ_F). If $[\Phi_T]_k$ and Φ_F are not orthogonal, then the algorithm reports that (Φ_T, Φ_F) has no k-convex extension. The claimed computational complexity follows from Theorem 3.10 and Lemma 2.4. □

Definition 4.3. The *convexity index* χ of a pdBf (Φ_T, Φ_F) is the maximum number k for which a k-convex extension of (Φ_T, Φ_F) exists.

Theorem 4.4. *Given a pdBf* (Φ_T, Φ_F), *we can find in* $O(n \log n|\Phi_T|(|\Phi_T| + |\Phi_F|))$ *time the convexity index* χ *of* (Φ_T, Φ_F), *and construct a prime DNF representation of its minimum* χ*-convex theory.*

Proof. Let us define the *convexity index algorithm*. It simply does a binary search for k on $[2, \ldots, n]$, checking each time whether the pdBf (Φ_T, Φ_F) has a k-convex extension by calling the k-convex theory algorithm described in the proof of Corollary 4.2. The claimed computational complexity follows from the fact that the binary search makes at most $O(\log n)$ steps. □

Theorem 4.5. *Given a pdBf* (Φ_T, Φ_F), *we can check in* $O(n|\Phi_T|(|\Phi_T| + |\Phi_F|)$*length* $(\Phi_T))$ *time whether it has a unique k-convex theory.*

Proof. Let the DNF $\bigvee_{i=1}^{s} P_i$ be the prime representation of the minimum k-convex theory h_k of the given pdBf, and assume that other k-convex theories exist. Let then f_u be another k-convex theory of (Φ_T, Φ_F) and $\bigvee_{j=1}^{t} Q_j$ be the prime representation of it. In view of our assumption, Theorem 4.2 shows that

$$\bigvee_{i=1}^{s} P_i \leqslant \bigvee_{j=1}^{t} Q_j. \tag{1}$$

We shall show that each P_i for $i \in [1, \ldots, s]$ is majorized by a Q_j for some $j \in [1, \ldots, t]$. Assume to the contrary that none of the Q_j's is a majorant of P_1. It follows from (1) that

$$P_1 = P_1Q_1 \vee P_1Q_2 \vee \cdots \vee P_1Q_t. \tag{2}$$

The right-hand side of (2) must contain at least two terms which are not identically zero, since otherwise the remaining Q_j would be a majorant of P_1. However, any pair of Q_j's conflicts in at least $k+1$ variables, making the application of the consensus method impossible, which contradicts the fact that P_1 is a prime implicant of the right hand side.

We can show in a similar way that every term Q_j for $j \in [1,\dots,t]$ is a majorant of a P_k for some $k \in [1,\dots,s]$. Assume to the contrary that Q_1 is not a majorant of any of the P_i's. Since f_u is a theory, Q_1 must cover a true point from Φ_T, say x. Let P_k be the term covering this true point. By the argument above, $P_k \leqslant Q_r$ for some $r \in [2,\dots,s]$. However, it is impossible that Q_1 and Q_r cover the same true point while being at a distance of at least $k+1$ from each other.

The algorithm, which will decide whether the pdBf (Φ_T, Φ_F) has a unique k-convex theory, first runs the k-convex theory algorithm on (Φ_T, Φ_F), and outputs "No" if $[\Phi_T]_k$ is not orthogonal to Φ_F. If $[\Phi_T]_k$ is orthogonal to Φ_F, then the algorithm examines one by one every occurrence of each literal in $[\Phi_T]_k$. Let l^P denote the occurrence of a literal l in a term P of $[\Phi_T]_k$. Let further $[\Phi_T]_k^{l^P}$ denote the DNF obtained from $[\Phi_T]_k$ by removing l from P. Then the algorithm runs the k-convexification method and checks whether $[[\Phi_T]_k^{l^P}]_k$ is orthogonal to Φ_F. If there exists at least one l^P such that the DNFs $[[\Phi_T]_k^{l^P}]_k$ and Φ_F are orthogonal, then the algorithm reports that (Φ_T, Φ_F) has more than one k-convex theory. Otherwise, the algorithm reports that $h_k = [\Phi_T]_k$ is the unique k-convex theory of (Φ_T, Φ_F).

The claimed computational complexity follows from the fact that $|h_k| \leqslant |\Phi_T|$ and $length(h_k) \leqslant length(\Phi_T)$. □

Example 4.6. Consider the pdBf (Φ_T, Φ_F) where

$$\Phi_T = x_1x_2x_3x_4x_5x_6x_7x_8 \vee x_1x_2x_3x_4x_5\bar{x}_6\bar{x}_7\bar{x}_8 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 \quad \text{and} \quad \Phi_F = \bar{x}_1x_3.$$

Application of the convexity index algorithm will yield $k=4$ and the minimum 4-convex theory $[\Phi]_4 = x_1x_2x_3x_4x_5 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5$. Note that Φ_T is already a 2-convex theory of (Φ_T, Φ_F). Note also that the above example shows the fact that there are more than one 2-convex theories of the given pdBf, namely Φ_T and $[\Phi]_4$ (the latter being 4-convex, it is also 3-convex and 2-convex). Note however that $[\Phi]_4$ is the unique 4-convex theory of the given pdBf.

Theorem 4.7. *Given a pdBf (Φ_T, Φ_F) and a point $\mathbf{x}$ which is not covered by $\Phi_T \vee \Phi_F$, we can check in*
- $O(n|\Phi_T|(|\Phi_T| + |\Phi_F|)$ *time if all the k-convex extensions of (Φ_T, Φ_F) agree in $\mathbf{x}$;*
- $O(n|\Phi_T|^2(|\Phi_T| + |\Phi_F|)$ *time if all the k-convex theories of (Φ_T, Φ_F) agree in $\mathbf{x}$.*

Proof. We check first whether all k-convex extensions of (Φ_T, Φ_F) take the value 1 in x. To do this, we run the k-convex theory algorithm on (Φ_T, Φ_F) and check whether the h_k constructed by the algorithm takes the value 1 in x. If yes, then every other

k-convex extension of (Φ_T, Φ_F) will also take the value 1 in x, since h_k is the smallest one.

Let us now assume that h_k takes the value 0 in x. Then we add the minterm of x to Φ_T, denote the resulting DNF by Φ'_T, and run the k-convex theory algorithm on (Φ'_T, Φ_F). If the algorithm fails to produce a k-convex extension, then all k-convex extensions of (Φ_T, Φ_F) take the value 0 in x. Let us now assume that the algorithm produces a k-convex function h'_k. Then the two k-convex extensions of (Φ_T, Φ_F), h_k and h'_k, disagree in x. If h'_k happens to be a theory of (Φ_T, Φ_F), then we already obtained two k-convex theories of (Φ_T, Φ_F) that disagree in x.

Let us now assume that h'_k is not a theory of (Φ_T, Φ_F), i.e. the term P of h'_k that covers x, does not cover any points in Φ_T. By construction, every k-convex theory of (Φ_T, Φ_F) that takes the value 1 in x must be a majorant of h'_k. Let h''_k be such a theory of (Φ_T, Φ_F). As was shown in the previous proof, every term in h'_k must be majorized by a term in h''_k. Let P'' be the term of h''_k such that $P'' \geqslant P$. Since h''_k is a k-convex theory of (Φ_T, Φ_F), its term P'' must cover some points in Φ_T, and therefore must majorize those terms of h'_k that also cover the same points in Φ_T. Therefore, there must exist another term $P' \neq P$ in h'_k such that $P'' \geqslant P'$. Since h''_k is a k-convex majorant of h'_k, it must be true that $P'' \geqslant [P, P']$. It follows that it is sufficient to try one by one every term $P' \neq P$, replace both P and P' in h'_k by $[P, P']$ resulting in the DNF h^*_k, and run the k-convex theory procedure on (h^*_k, Φ_F). If this procedure succeeds in producing a k-convex extension, then its output will be a theory of (Φ_T, Φ_F) that disagrees with h_k in x. Otherwise, another term P' should be tried. Eventually, when all the terms in h'_k are examined, we either construct a k-convex theory of (Φ_T, Φ_F) disagreeing with h_k in x, or prove that no such theory exists.

The claimed computational complexity follows from the fact that the number of terms in h'_k does not exceed $|\Phi_T|$. □

Example 4.8. For the pdBf given in Example 4.6, there is no agreement among the 3-convex theories in point $\boldsymbol{x} = (0,0,0,0,1,0,1,1)$. Indeed, one of the 3-convex theories, namely $x_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$, characterizes this point as true whereas $[\Phi]_4$ characterizes it as false.

5. Partially defined convex theories of PDBFs

The LAD problems we have considered in the previous sections of this paper focused on constructing extensions of the given data, i.e. fully defined Boolean functions. In many real life problems, certain Boolean vectors are not only absent from the given data, but are in fact infeasible, i.e. can never occur. For example, if the Boolean variable x_v takes the value 1 iff the systolic blood pressure is greater than or equal to some value v, and the Boolean variable y_v takes the value 1 iff the diastolic blood pressure is greater than or equal to v, then it is known that any observation in which $x_v = 0$ and $y_v = 1$ can never occur. In such situations a fully defined function may be inadequate as a model of the phenomenon, because the structural properties of feasible points may turn out to be too restrictive if applied to all the vectors in the Boolean hypercube.

In order to clarify this point, let us consider the main subject of this paper, i.e. the property of convexity. In the previous sections it was assumed that only the true points of the function possess the property of convexity, i.e. every pair of true points at distance at most k_1 is convexly connected. In many situations negative points may also possess the same convexity property, i.e. every pair of false points at distance at most k_0 is convexly connected. It was shown in [11] that there are only $n+2$ distinct fully defined Boolean functions of n variables with the property that both the set of true points and the set of false points are 2-convex. It is therefore natural not to limit the search for extensions of the given data set to these $n+2$ fully defined functions, but to also allow *partially defined* extensions as long as their true points and their false points both possess the desirable convexity properties. With this in mind, we introduce the following definitions.

A subset S of points in a Boolean hypercube defines naturally a Boolean function f_S whose set of true points is S. For the sake of brevity, we shall frequently denote f_S simply by S. For example, if f_S belongs to a class $\mathscr{C}$ of Boolean functions, we may write $S \in \mathscr{C}$.

Definition 5.1. Given a pdBf (T,F) and a pair of classes of Boolean functions $(\mathscr{C}_T, \mathscr{C}_F)$, a *partially defined extension* (pde) of (T,F) in $(\mathscr{C}_T,\mathscr{C}_F)$ is a pdBf (S_T,S_F) such that $T \subseteq S_T$, $F \subseteq S_F$, $S_T \cap S_F = \emptyset$, and $S_T \in \mathscr{C}_T$ and $S_F \in \mathscr{C}_F$.

Definition 5.2. Given a pdBf (T,F) and a pair of classes of Boolean functions $(\mathscr{C}_T, \mathscr{C}_F)$, a *partially defined theory* (pdt) of (T,F) in $(\mathscr{C}_T,\mathscr{C}_F)$ is a pde of (T,F) in $(\mathscr{C}_T,\mathscr{C}_F)$ for which there exists a pair of DNFs (Φ_T,Φ_F) such that Φ_T is a positive theory of (T,F) and Φ_F is a negative theory of (T,F).

The main problem to be studied in this section is the following. Given a pdBf (T,F) and a pair of classes of Boolean functions $(\mathscr{C}_T,\mathscr{C}_F)$, check whether a pdt of (T,F) in $(\mathscr{C}_T,\mathscr{C}_F)$ exists, and if yes, find one. More specifically, we will consider this problem for the case where $\mathscr{C}_T$ is the class of k_1-convex Boolean functions, and $\mathscr{C}_F$ is the class of k_0-convex Boolean functions, and any such pde will be said to belong to the class of (k_1,k_0)-*convex* pdBf's.

It follows from Proposition 3.2 that if a given pdBf has a (k_1,k_0)-convex pde, then it will also have a (k_1',k_0')-convex pde for any $k_1' \leqslant k_1$ and any $k_0' \leqslant k_0$. Let us call a pair of numbers (k_1,k_0) a *non-dominated* pair of a pdBf (T,F) if this pdBf has a (k_1,k_0)-convex pde, but does not have a (k_1+1,k_0)-convex pde or a (k_1,k_0+1)-convex pde. Clearly, the set of all non-dominated pairs describes completely the possible convex partially-defined extensions of the given pdBf.

Theorem 5.3. *Given a pdBf* (Φ_T,Φ_F), *all the non-dominated pairs of* (Φ_T,Φ_F) *can be generated in* $O(n^2(\max\{|\Phi_T|,|\Phi_F|\} + \log n \min\{|\Phi_T|,|\Phi_F|\})(|\Phi_T| + |\Phi_F|))$ *time.*

Proof. Let us assume that the given pdBf (Φ_T, Φ_F) has a k_1-convex extension, and h_{k_1} is the output of the k-convex theory algorithm. If (Φ'_T, Φ'_F) is a (k_1, k_0)-convex pde of (Φ_T, Φ_F), then (h_{k_1}, Φ'_F) is also a (k_1, k_0)-convex pde of (Φ_T, Φ_F) since h_{k_1} is a minorant of Φ'_T. Moreover, Φ'_F must also be a k_0-convex extension of the pdBf (Φ_F, h_{k_1}). Therefore, all non-dominated pairs of (Φ_T, Φ_F) can be obtained by examining one by one all the values $k \in \{2, \dots, n\}$ starting with 2 and determining for each k whether (Φ_T, Φ_F) has a k-convex extension by running the k-convex theory algorithm to obtain h_k, if any. Then for each such k that a k-convex extension exists, we can run on (Φ_F, h_k) the convexity index algorithm described in the proof of Theorem 4.4. Let us denote the result of this algorithm by $m(k)$.

Let us also denote by K the set of such values of k that $m(k) > m(k+1)$ for every $k \in K$. Then, clearly, the set of all non-dominated pairs is $\{(k, m(k)) | k \in K\}$. It follows from Theorems 4.2 and 4.4 that the described procedure has the computational complexity of $O(n^2(|\Phi_T| + \log n|\Phi_F|)(|\Phi_T| + |\Phi_F|))$.

The obtained computational complexity can be actually improved to

$$O(n^2(\max\{|\Phi_T|, |\Phi_F|\} + \log n \min\{|\Phi_T|, |\Phi_F|\})(|\Phi_T| + |\Phi_F|)).$$

This follows from the fact that all the non-dominated pairs of (Φ_T, Φ_F) can be obtained from the non-dominated pairs of (Φ_F, Φ_T) by simply swapping the numbers in each pair. Therefore, the algorithm presented above can be applied to (Φ_F, Φ_T) if $|\Phi_F| > |\Phi_T|$. □

Note that given a (k_1, k_0)-convex pde (h_1, h_0) of a pdBf (T, F) which has been constructed using the k-convex theory procedure, we can check easily whether this pde is in fact an extension of (T, F), i.e. if any unknown point can always be classified using (h_1, h_0). Indeed, since h_1 and h_0 are disjoint, the number of true points of $h_1 \vee h_0$ is simply the sum of the true points of h_1 and those of h_0. On the other hand, both h_1 and h_0 are constructed as orthogonal DNFs, and therefore the number of their true points is easily computable, as described in the proof of Proposition 2.5. Clearly, (h_1, h_0) is an extension of (T, F) iff the total number of true points of $h_1 \vee h_0$ equals 2^n.

If our objective is to build a question-asking strategy, we can easily exhibit a point not covered by either h_1 or h_0, if any. Indeed, we simply apply Proposition 2.5 to the orthogonal DNF $h_1 \vee h_0$. Additionally, using Proposition 2.5 we can output all such unclassified points in time polynomial in their total number.

6. Probabilistic properties

In order to analyze the predictive performance of our algorithms on random data, we shall follow the *probably approximately correct* (PAC) model of computational learning theory (see e.g. [1, 5, 13]), which assumes that data points are generated randomly according to a fixed unknown probability distribution on B^n, and that they are classified by some unknown Boolean function f belonging to a class $\mathscr{C}(n)$ of Boolean functions.

The class $\mathscr{C}(n)$ is called *PAC-learnable* if for any $\varepsilon, \delta \in (0,1)$, one can draw randomly a polynomial number of points $Poly(n, \frac{1}{\varepsilon}, \frac{1}{\delta})$ from B^n together with their classifications and can find in polynomial time a function $g \in \mathscr{C}(n)$ such that

$$Prob(Prob(f \Delta g) > \varepsilon) < \delta. \tag{3}$$

Here ε is called the *accuracy* of g, δ is called the *confidence* in this accuracy, and $f \Delta g$ denotes the set of Boolean vectors where f and g disagree.

Given a set $S \subseteq B^n$, let us denote by $cl_{\mathscr{C}(n)}(S)$ the number of different dichotomies induced on S by the functions in $\mathscr{C}(n)$.

Definition 6.1. The *Vapnik–Chervonenkis dimension* (VC-dimension) of a set of Boolean functions $\mathscr{C}(n)$ is the largest integer $d(\mathscr{C}(n))$ such that there exists a set $S \subseteq B^n$ of cardinality $d(\mathscr{C}(n))$ for which $cl_{\mathscr{C}(n)}(S) = 2^{d(\mathscr{C}(n))}$.

Clearly, for any class $\mathscr{C}(n)$ of Boolean functions of n variables, $d(\mathscr{C}(n)) \leqslant 2^n$. It is also well known that

$$d(\mathscr{C}(n)) \leqslant \log_2 |\mathscr{C}(n)|. \tag{4}$$

Let $\mathscr{F}(n,k)$ denote the class of k-convex functions of n Boolean variables.

Theorem 6.2. *The VC-dimension of the class $\mathscr{F}(n,k)$ has the following lower bound*:

$$d(\mathscr{F}(n,k)) \geqslant \frac{2^n}{\sum_{i=0}^{k} \binom{n}{i}}.$$

Proof. Let M be a largest set of Boolean vectors of dimension n whose pairwise Hamming distances are at least $k+1$, and let m be the cardinality of M. If we take balls of radius k around each point in M, then $\sum_{i=0}^{k} \binom{n}{i}$ is the number of points that fall within each ball, and each point in the hypercube must be inside at least one of these balls. Therefore,

$$m \sum_{i=0}^{k} \binom{n}{i} \geqslant 2^n.$$

If we have a set of vectors which are pairwise at distance at least $k+1$, then any subset of this set can define the set of true points of a k-convex Boolean function. □

Theorem 6.2 and inequality (4) imply

Corollary 6.3. *The number of functions in the class $\mathscr{F}(n,k)$ is at least*

$$|\mathscr{F}(n,k)| \geqslant 2^{\frac{2^n}{\sum_{i=0}^{k} \binom{n}{i}}}.$$

In the following we use the notation $\phi = \Omega(\psi)$ to denote that there exists a constant c such that $\phi \geqslant c\psi$.

Corollary 6.4. *If* $k \leqslant \frac{n}{2} - \Omega(n)$, *then*
1. $d(\mathscr{F}(n,k)) \geqslant 2^{\Omega(n)}$;
2. $|\mathscr{F}(n,k)| \geqslant 2^{2^{\Omega(n)}}$.

Proof. Using Chernoff's bound [7]

$$\sum_{i<\frac{n}{2}-m} \binom{n}{i} < 2^n e^{-\frac{2m^2}{n}} \quad \text{for } 0 \leqslant m \leqslant \frac{n}{2},$$

if we let $k = \frac{n}{2} - m + 1$, then

$$d(\mathscr{F}(n,k)) \geqslant \frac{2^n}{\sum_{i=0}^{k} \binom{n}{i}} > \frac{2^n}{2^n e^{-\frac{2m^2}{n}}} = e^{\frac{2m^2}{n}} \geqslant e^{\Omega(n)} \quad \text{if } m = \Omega(n). \qquad \Box$$

Lemma 6.5. *The number of functions in the class* $\mathscr{F}(n,k)$ *is bounded in the following way*:

$$|\mathscr{F}(n,k)| \leqslant 3^{k2^{n-k}}.$$

Proof. It is known that $\mathscr{F}(n,n)$ represents the class of monomials, and that there exist exactly 3^n different monomials of n variables. If we fix a set of $n-k$ variables in an arbitrary k-convex function, the resulting function must be a monomial since it should still be a k-convex function. Since the $n-k$ variables can be fixed in 2^{n-k} different ways, and since each such restriction can yield at most 3^k different functions, we must have $|\mathscr{F}(n,k)| \leqslant (3^k)^{2^{n-k}}$. $\Box$

The upper bound on the number of k-convex functions obtained in Lemma 6.5 is not sharp enough to imply PAC-learnability results for k close to $n/2$. To obtain a sharper bound, we need to estimate first how many prime implicants a k-convex function can have when k is large.

Lemma 6.6. *If* $k > \frac{n}{2} - 1$, *then the number of prime implicants of a function in* $\mathscr{F}(n,k)$ *cannot exceed* $2(k+1)/[2(k+1)-n]$.

Proof. Let $\{T_1, T_2, \ldots, T_s\}$ be the prime implicants of $f \in \mathscr{F}(n,k)$; let $e_i(j,l) = 1$ if T_j and T_l conflict in x_i and let $e_i(j,l) = 0$ otherwise. It follows from Proposition 3.2 that for any pair j, l

$$\sum_{i=1}^{n} e_i(j,l) \geqslant k+1. \qquad (5)$$

Summing up these inequalities for all pairs j, l, we get

$$\sum_{j=1}^{s-1} \sum_{l=j+1}^{s} \sum_{i=1}^{n} e_i(j,l) \geqslant (k+1)\frac{s(s-1)}{2}. \tag{6}$$

Let us consider the graph G_i whose vertices are the terms $\{T_1, T_2, \dots, T_s\}$, and where T_j and T_l are connected if and only if they conflict in x_i. This graph is bipartite: the terms containing x_i belong to the first part, the terms containing $\bar{x}_i$ belong to the second part, and the terms not involving x_i can be considered as belonging to either part. Since G_i is bipartite, the number of edges in it is limited by $\frac{s^2}{4}$. Therefore,

$$\sum_{j=1}^{s-1} \sum_{l=j+1}^{s} e_i(j,l) \leqslant \frac{s^2}{4}. \tag{7}$$

Since $\sum_{j=1}^{s-1} \sum_{l=j+1}^{s} \sum_{i=1}^{n} e_i(j,l) = \sum_{i=1}^{n} \sum_{j=1}^{s-1} \sum_{l=j+1}^{s} e_i(j,l)$, (6) and (7) imply

$$n\frac{s^2}{4} \geqslant (k+1)\frac{s(s-1)}{2},$$

or equivalently,

$$s(2(k+1)-n) \leqslant 2(k+1). \tag{8}$$

If $k > \frac{n}{2} - 1$, then (8) implies the claim of the lemma. □

Corollary 6.7. *The number of prime implicants of any function in $\mathscr{F}(n,k)$ does not exceed*
1. $n+1$ *if* $k \geqslant \frac{n}{2} - \frac{1}{2}$,
2. $1 + \frac{1}{c}$ *if* $k \geqslant \frac{n}{2} - 1 + cn$.

Remark that since k is integer, the inequality $k > \frac{n}{2} - 1$ implies $k \geqslant \frac{n}{2} - \frac{1}{2}$.

Since the number of different terms in n Boolean variables is 3^n, Lemma 6.6 and Corollary 6.7 imply

Corollary 6.8. *The number of functions in $\mathscr{F}(n,k)$ does not exceed*
1. $3^{\frac{2n(k+1)}{2(k+1)-n}} \leqslant 3^{n(n+1)}$ *if* $k > \frac{n}{2} - 1$,
2. $2^{O(n)}$ *if* $k \geqslant \frac{n}{2} + \Omega(n)$.

The results obtained above allow us to characterized the PAC-learnability of the classes of k-convex functions using upper and lower bounds on the sample complexity of PAC-learning algorithms.

The following result was shown in [10] (see also [1, 13]).

Proposition 6.9. *Any PAC-learning algorithm for a class $\mathscr{C}(n)$ may need to draw a random sample of at least $\Omega(\frac{d(\mathscr{C}(n))}{\varepsilon})$ points to satisfy the PAC-learning condition* (3).

The following result is well known in computational learning theory (see e.g. [1, 5, 13]).

Proposition 6.10. *Any Boolean function* $g \in \mathscr{C}(n)$ *which correctly classifies a random sample of* $\Omega(\frac{1}{\varepsilon}\log\frac{|\mathscr{C}(n)|}{\delta})$ *points satisfies the PAC-learning condition* (3).

Theorem 6.11. 1. *If* $k \leqslant \frac{n}{2} - \Omega(n)$, *then the class* $\mathscr{F}(n,k)$ *is not PAC-learnable.*
2. *If* $k > \frac{n}{2} - 1$, *then the class* $\mathscr{F}(n,k)$ *is PAC-learnable.*

Proof. The first statement follows from Proposition 6.9 and Corollary 6.4(1) showing that a random sample of exponential size is needed to satisfy the PAC-learning condition (3).

To prove the second statement, note that it follows from Proposition 6.10 and Corollary 4.2 that the k-convex theory procedure provides a polynomial PAC-learning algorithm when it is applied to a random sample of size $\Theta(\frac{1}{\varepsilon}\log\frac{|\mathscr{F}(n,k)|}{\delta})$ [1] (which is polynomial in n by Corollary 6.8(1)). □

Corollary 6.12. *The class* $\bigcup_{k=\lceil\frac{n-1}{2}\rceil}^{n}\mathscr{F}(n,k)$ *is PAC-learnable.*

Proof. Clearly,

$$\left|\bigcup_{k=\lceil\frac{n-1}{2}\rceil}^{n}\mathscr{F}(n,k)\right| \leqslant \left\lceil\frac{n+1}{2}\right\rceil 3^{n(n+1)}.$$

It is therefore sufficient to draw a random sample of $\Theta(\frac{1}{\varepsilon}(n^2+\log\frac{n}{\delta}))$ points and apply the k-convex theory procedure only for $k=\lceil\frac{n-1}{2}\rceil$. □

Remark 6.13. The application of the k-convex theory procedure in the PAC-learning framework can be somewhat simplified. More exactly, in view of the assumed k-convexity of the unknown function, the k-convex hull $[\Phi_T]_k$ is known a priori to be orthogonal to $[\Phi_F]$, i.e. this property does not have to be checked during the execution of the algorithm. Therefore, for $k > \frac{n}{2} - 1$, k-convex functions are PAC-learnable from positive examples only.

7. Concluding remarks

We have studied in this paper convexity properties of partially-defined Boolean functions. A polynomial-time algorithm was developed for recognizing those pdBfs which admit k-convex extensions. As a by-product of this algorithm, a k-convex theory of a pdBf was constructed when possible. It was shown how to determine in polynomial time the convexity index of a given pdBf, and how to check in polynomial time

[1] Here $\phi = \Theta(\psi)$ denotes that $\phi = O(\psi)$ and $\phi = \Omega(\psi)$ simultaneously.

whether a given pdBf has a unique k-convex theory. Finally, a polynomial algorithm was presented to establish whether all k-convex extensions and/or all k-convex theories of a given pdBf (T,F) take the same value in a point not in $T \cup F$.

It is easy to see that if f is a k-convex function and α is one of its true points, then there exists a unique prime implicant of f which takes the value 1 in α. This property defines the class of Boolean functions which we call *orthogonal*. It can be checked easily that the class of orthogonal functions is characterized by the property that every two prime implicants conflict in at least two literals. The class of k-convex functions (for any $k \geqslant 2$) is obviously included in the class of orthogonal functions, and this inclusion is proper (e.g. the function $xy \vee \overline{x}\overline{y}$ is orthogonal but not k-convex).

It is easy to check that all the results presented in this paper for k-convex functions have straightforward extensions to the case of orthogonal functions. To see this, in most cases it is sufficient to formally use $k=1$ in the stated results and algorithms. We note however, that every Boolean function would satisfy Definition 3.1 for $k=1$.

Acknowledgements

The authors extend their thanks for the partial support provided by the Office of Naval Research (grant N00014-92-J1375) and the National Science Foundation (grant NSF-DMS-9806389). The authors also express their gratitude to Endre Boros and Dan Roth for useful discussions, and to the anonymous referee for careful analysis and helpful suggestions.

References

[1] M. Anthony, N. Biggs, Computational Learning Theory, Cambridge University Press, Cambridge, 1992.
[2] M.O. Ball, G.L. Nemhauser, Matroids and a reliability analysis problem, Math. Oper. Res. 4 (1979) 132–143.
[3] A. Beimel, F. Bergadano, N.H. Bshouty, E. Kushilevitz, S. Varricchio, On the applications of multiplicity automata in learning, in: Proc. 37th Annual IEEE Symp. on Foundations of Computer Science (FOCS'1996), pp. 349–358.
[4] A. Blake, Canonical expressions in Boolean algebra, Ph.D. Thesis, University of Chicago, August 1937.
[5] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Learnability and the Vapnik–Chervonenkis dimension, J. Assoc. Comput. Machinery 36 (4) (1989) 929–965.
[6] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik, An implementation of logical analysis of data, IEEE Transactions on Knowledge and Data Engineering, accepted for publication.
[7] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, Ann. Math. Statist. 23 (1952) 493–509.
[8] C.J. Colbourn, The Combinatorics of Network Reliability, Oxford University Press, New York, 1987.
[9] Y. Crama, P.L. Hammer, T. Ibaraki, Cause-effect relationships and partially defined Boolean functions, Ann. Oper. Res. 16 (1988) 299–326.
[10] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant, A general lower bound on the number of examples needed for learning, Inform. Comput. 82 (3) (1989) 247–261.

[11] O. Ekin, P.L. Hammer, A. Kogan, On connected Boolean functions, Discrete Appl. Math., accepted for publication.
[12] P.L. Hammer, S. Rudeanu, Boolean Methods in Operations Research, Springer, Berlin, 1968.
[13] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, MIT Press, Cambridge, MA, 1994.
[14] W. Quine, A way to simplify truth functions, Amer. Math. Monthly 62 (1955) 627–631.
[15] K.G. Ramamurthy, Coherent Structures and Simple Games, Kluwer Academic Publishers, Dordrecht, 1990.