Dualisation, Decision Lists and Identification of Monotone Discrete Functions

Jan C. Bioch
Department of Computer Science
Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam
The Netherlands
bioch@few.eur.nl

March, 1998

Abstract

Many data-analysis algorithms in machine learning, datamining and a variety of other disciplines essentially operate on discrete multi-attribute data sets. By means of discretisation or binarisation also numerical data sets can be successfully analysed. Therefore, in this paper we view/introduce the theory of (partially defined) discrete functions as an important theoretical tool for the analysis of multi-attribute data sets. In particular we study monotone (partially defined) discrete functions. Compared with the theory of Boolean functions relatively little is known about (partially defined) monotone discrete functions. It appears that decision lists are useful for the representation of monotone discrete functions. Since dualisation is an important tool in the theory of (monotone) Boolean functions, we study the interpretation and properties of the dual of a (monotone) binary or discrete function. We also introduce the dual of a pseudo-Boolean function. The results are used to investigate extensions of partially defined monotone discrete functions and the identification of monotone discrete functions. In particular we present a polynomial time algorithm for the identification of so-called stable discrete functions.

Keywords: Monotone discrete functions, decision list representations, partially defined discrete functions, dualisation, pseudo-Boolean functions, stable functions and identification of monotone discrete functions.

1 Introduction

Real-world datasets analysed in machine learning, datamining and decision sciences very often contain multi-attribute descriptions of objects. Some of these attributes may represent decision or classification variables. The algorithms used to find an explanation of the dataset, or to discover association rules, or to induce concepts from positive and negative examples, or to induce decision trees, are essentially based on Boolean or discrete attributes. Successful applications of datasets with numerical attributes are made possible by the process of discretisation or binarisation before or during the analysis of the data. Recent successful methodologies such as logical analysis of data (LAD) [11, 17] and association rules (AR) [3], that also can handle numerical data, are even based on Boolean

attributes. Since many of these data-analysis algorithms essentially operate on discrete multi-attribute datasets we introduce discrete functions as an important theoretical tool for the analysis of these data sets.

In this paper we study monotone discrete functions as a tool for ordinal classification and exact learning (identification), and as a basis for multi-attribute decision making. However, contrary to Boolean functions, very little is known about monotone discrete functions, see Davio et al. [13] and Störmer [21].

In ordinal classification the attributes and classes are linearly ordered sets. In monotone classification problems the classification rules are supposed to be order preserving maps. In fact, if the data set D is a subset of a partially ordered set X and Y is a finite linear ordered set, then a monotone classification rule is an order preserving map $F : X \mapsto Y$. Let f be the monotone function $f : D \mapsto Y$ that represent the class labeling of the data elements in D. Then F is called an extension of f if F agrees with f on D: $F(x) = f(x)$, $\forall x \in D$. Thus in ordinal classification problems one is interested in extensions, such as monotone decision trees, minimal(maximal) extensions etc. Ordinal classification for multi-attribute decision making has recently been studied by Ben-David [4, 5], Makino and Ibaraki [18], Potharst and Bioch [7, 19]. The theory of extensions of partially defined Boolean functions has been extensively studied by Boros, Ibaraki and Makino [9, 10], and in the framework of LAD by Boros, Crama, Ibaraki, Hammer and Kogan [12, 11, 17]. From a theoretical point of view multi-attribute ordinal classification is essentially the study of monotone extensions of partially defined discrete functions (pdDfs). The only paper we know of on this subject is of Greco et al. [15, 16]. Although this paper is informal, it describes an interesting problem on risk analysis and discusses the idea of what can be viewed as the discrete version of a (Boolean) pattern which plays a central rôle in the LAD methodology.

The problem of the identification of a given monotone discrete function f discussed in this paper is in fact a special case of the extension problem discussed above. In this case the data set is not fixed, but extended by asking queries to an oracle about the values of f in specific points of the input space X, until only one (monotone) extension is possible (the given function). In the case f is a Boolean function, in computational learning theory this is called exact learning of a theory f by asking membership queries only [1, 2]. However, in this paper we do not only acquire explicit knowledge about the set of minimal vectors of the monotone discrete function f but also of the set of maximal vectors. The identification problem (in this sense) for Boolean functions is studied by Bioch and Ibaraki in [6]. It appears that this problem is polynomially equivalent to many other interesting problems. The complexity of this problem, that is related to that of dualisation is still not known, although Fredman and Khachiyan [14] have proved a result on the mutual duality of positive Boolean functions that tells that the complexity of identification discussed here is unlikely to be NP-hard.

This paper is organised as follows. In section 2 we discuss representations of monotone discrete functions, such as normal forms (DNF and CNF). It is known [21] that monotone discrete functions have a unique disjunctive normal form (DNF) consisting of all prime implicants. As far as we know this is essentially all that is known about monotone discrete functions. We present this result by using discrete variables introduced in [13] so that the correspondence to the theory of (partially defined) Boolean functions becomes clear. In this section we also introduce decision lists and binary monotone level functions as a convenient representation of monotone functions. In section 3 we introduce the dual of

a discrete function and study its properties and interpretation. We also introduce the dual of a pseudo-Boolean function and we argue that a pseudo-Boolean function can be viewed as a discrete function. It appears that dualisation of a monotone discrete function is equivalent to that of its binary level functions. Monotone extensions of partially defined monotone discrete functions are discussed in section 4. Finally, in section 5 we discuss the problem of identification of a monotone discrete function f. This problem can be reduced to the identification of the set of monotone binary level functions of f. This shows that the complexity of dualisation and identification of monotone discrete functions is not essentially different from that of monotone Boolean functions. We also introduce the class of monotone stable functions and present an algorithm for the identification of monotone discrete functions. We show that for a stable function f the running time of this algorithm is quadratic in the size of f and its dual.

2 Representations of positive discrete functions

In this paper we study discrete functions of the form

$$f : X_1 \times X_2 \times \ldots \times X_n \to Y, \tag{1}$$

where $X = X_1 \times X_2 \times \ldots \times X_n$ and Y are finite sets. As known [21], without loss of generality we may assume: $X_i = \{0, 1, \ldots, n_i\}$ and $Y = \{0, 1, \ldots, m_i\}$. For example, if $Y = \{\alpha_0, \alpha_1, \ldots, \alpha_m | \ \alpha_i \in \mathbb{R}\}$ then we we will replace Y by $\{0, 1, \ldots, m\}$. If $|Y| = 2$, then f is called a *binary* function. A discrete function is called a *logic* function if $|X_i| = |Y|$ for $1 \leq i \leq n$. A Boolean function f is a binary logic function: $f : \{0,1\}^n \to \{0,1\}$. As a general reference for discrete functions we mention [13, 21]. The set X becomes a partially ordered set if we define $\forall x, y : x \leq y \Leftrightarrow x_i \leq y_i, 1 \leq i \leq n$. Furthermore, we define the lub $\vee$ and the glb $\wedge$ of $x, y \in X$ as follows:

$$\begin{array}{l} x \vee y = v, \text{ where } v_i = \max\{x_i, y_i\} \\ x \wedge y = w, \text{ where } w_i = \min\{x_i, y_i\}. \end{array} \tag{2}$$

Therefore, we can consider X (and Y) as distributive lattices. The greatest and smallest element of X are respectively denoted by x^* and 0^*. (Quasi-)complementation for X is defined as: $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m)$, where $\overline{x_i} = n_i \Leftrightarrow x_i$. Similarly, the complement of $j \in Y$ is defined as $\overline{j} = m \Leftrightarrow j$. The following well-known properties of complementation hold:

$$\begin{array}{rcl} \overline{x \vee y} & = & \overline{x} \wedge \overline{y} \\ \overline{x \wedge y} & = & \overline{x} \vee \overline{y}. \end{array} \tag{3}$$

Note, that X is not a complemented lattice because e.g. $x \wedge \overline{x}$ is in general not equal to 0^*. The collection of all discrete functions $\mathcal{D}(X, Y) = \{f : X \to Y\}$ is partially ordered by

$$f \leq g \ \Leftrightarrow \ \forall x : f(x) \leq g(x). \tag{4}$$

Furthermore, $\{\mathcal{D}, \wedge, \vee\}$ is a distributive lattice if we define the lub $\vee$ and the glb $\wedge$ by respectively:

$$\begin{array}{l} (f \vee g)(x) = \max\{f(x), g(x)\} \\ (f \wedge g)(x) = \min\{f(x), g(x)\}. \end{array} \tag{5}$$

Finally, the complement of a discrete function f is defined by: $\overline{f}(x) = \overline{f(x)}$, and we have

$$\begin{array}{l} \overline{f \vee g} = \overline{f} \wedge \overline{g} \\ \overline{f \wedge g} = \overline{f} \vee \overline{g}. \end{array} \tag{6}$$

A discrete function f is called *positive* (monotone non-decreasing) if $x \leq y$ implies $f(x) \leq f(y)$. If f and g are positive functions, then $f \vee g$ and $f \wedge g$ are also positive. Therefore, the class of positive discrete functions is also a distributive lattice.

Notation We frequently use the following notation:
$(m] = \{j|\ 1 \leq j \leq m\}$ and $[m) = \{i|\ 0 \leq i \leq m \Leftrightarrow 1\}$.

Class functions and level functions

A discrete function f induces a partitioning on X with $m+1$ classes: $T_j(f) = \{x|f(x) = j\}$, $j \in [m+1)$. Elements of $T_j(f)$ are called *true vectors* of class j, or true vectors of the *class(indicator)* function f_j defined by:

$$f_j(x) = \begin{cases} 1 \text{ if } f(x) = j \\ 0 \text{ otherwise.} \end{cases} \tag{7}$$

Obviously, every discrete function is determined by its class (indicator) functions $f_j, j \in [m+1)$. However, class functions are not monotone.

A positive discrete function f can also be represented by m binary *monotone* functions. These functions called here the *level* functions of f are defined as follows:

$$\lambda_j(f)(x) = \begin{cases} 1 \text{ if } f(x) \geq j \\ 0 \text{ otherwise} \end{cases} \quad \textit{and} \quad \mu_i(f)(x) = \begin{cases} 1 \text{ if } f(x) \leq i \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

where $j \in (m]$ and $i \in [m)$.

Thus $\lambda_j(f) = \bigvee_{i=j}^{m} f_j$, $\lambda_m(f) = f_m$, and $\mu_i(f) = \bigvee_{j=0}^{i} f_j$, $\mu_0 = f_0$. From (8) it follows that:

- $\lambda_j(f)$ is a positive function, and $\mu_i(f)$ is a negative function
- $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_m$ and $\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{m-1}$.
- $f_j = \lambda_j \wedge \mu_j = \lambda_j \wedge \overline{\lambda}_{j+1} = \mu_j \wedge \overline{\mu}_{j-1},\ j \in (m \Leftrightarrow 1]$.
- $\mu_j = \overline{\lambda}_{j+1},\ j \in [m)$.

Lemma 1 *Let f and g be positive discrete functions.*
Then $f \leq g \Leftrightarrow \lambda_j(f) \leq \lambda_j(g),\ \forall j \in (m]$.

Proof Since the level functions are binary we have:

$$\lambda_j(f) \leq \lambda_j(g) \Leftrightarrow (\lambda_j(f) = 1 \Rightarrow \lambda_j(g) = 1). \tag{9}$$

Suppose $\lambda_j(f) \leq \lambda_j(g)$ and $f(x) = j$. Then $\lambda_j(f) = 1$, implying that $\lambda_j(g) = 1$. Therefore, $g(x) \geq j = f(x)$. Conversely, suppose $f \leq g$ and $\lambda_j(f)(x) = 1$.
Then $g(x) \geq f(x) \geq j$. Thus $\lambda_j(g)(x) = 1$. Therefore, $f \leq g \Rightarrow \lambda_j(f) \leq \lambda_j(g)$. □

Corollary 1 *If f and g are positive functions, then $f = g \Leftrightarrow \forall j \in (m]:\ \lambda_j(f) = \lambda_j(g)$.*

2.1 Monotone decision lists

In this subsection we discuss representations of positive discrete functions. It appears that positive functions can be conveniently represented by two kinds of decision lists: the maxlist and the minlist representation. To see this, we generalise the idea of a decision list well-known in the theory of Boolean functions, see e.g. [2]. Given a discrete function $f : X \mapsto Y$, binary test functions $t_j : X \mapsto \{0,1\}$, $1 \leq j \leq k \leq m$, and $j_0, \cdots, j_k \in Y$. Then the decision list representation of f, denoted by:

$$f = ((t_1, j_1);\ (t_2, j_2); \ldots ;\ (t_k, j_k);\ j_0)$$

is defined by:

if $t_1(x) = 1$ then set $f(x) = j_1$
  else if $t_2(x) = 1$ then set $f(x) = j_2$
    $\vdots$
    else if $t_k(x) = 1$ then set $f(x) = j_k$
      else set $f(x) = j_0$.

In the sequel we use two special types of decision lists for a positive function f based on the minimal and maximal vectors of f. A minimal vector v of class j is a vector such that $f(v) = j$ and no vector strictly smaller than v is also in $T_j(f)$. Similarly, a maximal vector w is a vector maximal in the class $T_j(f)$, where $j = f(w)$. Let $\min T_j(f), j \in (m]$ denote the set of minimal vectors of class j, and let $\max T_i(f)$, $i \in [m)$, denote the set of maximal vectors of class i. Then f is completely defined by each one of the sets $\min T(f)$ and $\max T(f)$, where:

$$\min T(f) = \bigcup_{j \in (m]} \min T_j(f) \text{ and } \max T(f) = \bigcup_{i \in [m)} \max T_i(f).$$

In the minlist representation $\min T(f)$ is used to test whether an input vector belongs to respectively class $m, m \Leftrightarrow 1, \ldots, 1$. Similarly, $\max T(f)$ is used in a maxlist representation of f to test whether an input vector belongs respectively to class $0, 1, \ldots, m \Leftrightarrow 1$.

Example 1 Let f be the positive ternary function $f : \{0,1,2\}^3 \mapsto \{0,1,2\}$ defined by the sets $\min T_1(f) = \{002, 200\}$ and $\min T_2(f) = \{210\}$, see table 1.

$\min T(f)$	$\max T(f)$	f
210		2
002, 200	122, 202	1
	021	0

Table 1: Representing f

The *minlist* representation of f is given by

$f(x) =$ if $x \geq 210$ then 2
  else if $x \geq 002, 200$ then 1
    else 0.

Note, that in this list $x \geq 002, 200$ is a shorthand for: $x \geq 002$ or $x \geq 200$. Given the set $\min T(f)$ we can compute $\max T(f)$, also given in table 1, by dualisation, see section 3. From this we obtain the decision list, called the *maxlist* of f:

$$\begin{aligned} f(x) = \text{if } x \leq 021 \text{ then } 0 \\ \text{else if } x \leq 122, 202 \text{ then } 1 \\ \text{else } 2. \end{aligned}$$

In general, given sets $\min_j$, $j \in (m]$, where each $\min_j \subset X$ is a set of incomparable vectors such that if $u \in \min_i$ and $v \in \min_j$, $i > j$, then $u \not\leq v$. Then the following function is positive and $\min T_j(f) = \min_j$.

$$\begin{aligned} f(x) = \text{if } x \in \uparrow\min_m \text{ then } m \\ \text{else if } x \in \uparrow\min_{m-1} \text{ then } m \Leftrightarrow 1 \\ \vdots \\ \text{else if } x \in \uparrow\min_1 \text{ then } 1 \\ \text{else } 0. \end{aligned}$$

Note, that if $M \subset X$, then the *upset* of M is defined by $\uparrow M = \{y \in X | \ \exists x \in M$ such that $x \leq y$. Similarly, the *downset* of M is defined as: $\downarrow M = \{x \in X | \ \exists y \in M$, where $x \leq y$. In the same way one can define a positive function by a maxlist, given sets $\max_i$, $i \in [m)$, where each $\max_i \subset X$ is a set of incomparable set of vectors such that $u \not\leq v$, whenever $u \in \max_i$ and $v \in \max_j$ and $i > j$. Here, the if-statements are of the form: if $x \in \downarrow\max_i$ then i, for $i \in [m)$.

2.2 Normal forms of positive discrete functions

It is well-known that positive Boolean functions have a unique disjunctive normal form (DNF) consisting of all prime implicants, and a unique conjunctive normal form (CNF) consisting of all prime implicates. It is known [21], that similar results hold in the case of positive discrete functions. We will briefly discuss the most important notions using our notation (2)-(7), and the decision list representation. See also Davio et al. [13] and Störmer [21].

Although the notations of prime implicants and prime implicates, well-known in Boolean function theory, can be generalised in several ways, for positive discrete functions these definitions coincide. Before presenting the normal forms for positive functions we define the following basic functions, called here the (discrete) *cubic* and *anti-cubic* function:

$$\begin{aligned} c_{v,j}(x) &= \text{if } x \geq v \text{ then } j \text{ else } 0, \quad j \in (m] \\ a_{w,i}(x) &= \text{if } x \leq w \text{ then } i \text{ else } m, \ i \in [m). \end{aligned}$$

A cubic function $c_{v,j}$ is called a *prime implicant* of f if $c_{v,j} \leq f$ and $c_{v,j}$ is maximal w.r.t. this property. Dually, $a_{w,i}$ is called a *prime implicate* of f if $f \leq a_{w,i}$ and $a_{w,i}$ is minimal w.r.t. this property.

In our notation (cf. Störmer [21]), the DNF of f:

$$f = \bigvee_{v,j} \{c_{v,j} \mid v \in \min T_j(f), \ j \in (m]\}, \tag{10}$$

is a unique representation of f as a disjunction of all its prime implicants. A similar result holds for the conjunctive normal form:

$$f = \bigwedge_{w,i} \{a_{w,i} \mid w \in \max T_i(f),\ i \in [m)\}. \tag{11}$$

Thus, the CNF given here is the unique representation of f as a conjunction of all prime implicates. We can write $c_{v,j}(a_{w,i})$ as a conjunction (disjunction) of discrete variables, introduced in [13]:

$$x_{ip} = \text{if } x_i \geq p \text{ then } m \text{ else } 0, \text{ where } 1 \leq p \leq n_i,\ i \in (n]. \tag{12}$$

Furthermore, we define $x_{in_i+1} = 0$. Thus we can write:

$$c_{v,j} = j.x_{1v_1}x_{2v_2}\cdots x_{nv_n} \tag{13}$$

$$a_{w,i} = i \vee x_{1w_1+1} \vee x_{2w_2+1} \cdots \vee x_{nw_n+1}. \tag{14}$$

Note here, that $j.x_{ip}$ denotes the conjunction $j \wedge x_{ip}$, where $j \in Y$ is a constant, and $x_{ip}x_{jq}$ denotes $x_{ip} \wedge x_{iq}$. Moreover, $m.x_{ip} = x_{ip}$, and if $p \leq q$, then $x_{ip}x_{iq} = x_{ip}$, $x_{ip} \vee x_{iq} = x_{iq}$.

Example 2 Let f be defined by the decision list:

$f(x) =$ if $x \geq 202$ then 3
else if $x \geq 200$ then 2
else if $x \geq 001, 110$ then 1
else 0.

Then $f(x) = (3).x_{12}x_{32} \vee 2.x_{12} \vee 1.x_{11}x_{21} \vee 1.x_{31}$. □

Using decision lists it is easy to see that every positive discrete function admits a DNF in which all discrete variables are positive. By using the complements of these variables it is also easy to see that every discrete function has a DNF expressed by positive and negative literals.

$$f(x) = \bigvee_{v,j} \{j.x_{v_1}x_{v_2}\cdots x_{v_n}\overline{x}_{1v_1+1}\overline{x}_{2v_2+1}\cdots\overline{x}_{nv_n+1} \mid f(v) = j\}. \tag{15}$$

Note, that if $p+1 \leq n_i$, $\overline{x}_{ip+1}$ means: if $x_i \geq p+1$ then 0 else m. Thus:

$$\overline{x}_{ip+1} = \text{if } x_i \leq p \text{ then } m \text{ else } 0. \tag{16}$$

Furthermore, we define $\overline{x}_{in_i+1} = m$.

3 Dualisation of discrete functions

The *dual* of a Boolean function f is defined as

$$f^d(x) = \overline{f}(\overline{x}). \tag{17}$$

Dualisation and the complexity of it is extensively studied in the theory of Boolean functions, see e.g. [6, 14]. In this paper we introduce the dual of a discrete functon using

the same definition. However, since we have generalised the concepts of complement, disjunction and conjunction to the lattice of discrete functions, not all properties necessarily hold. The following well-known properties remain valid:

$$\begin{aligned} f \leq g &\Leftrightarrow g^d \leq f^d \\ (f \vee g)^d &= f^d \vee g^d \\ (f \wedge g)^d &= f^d \vee g^d \\ (\overline{f})^d &= \overline{f}^d. \end{aligned}$$

Furthermore, if x_{ip} is a discrete variable and $j \in Y$ a constant then

$$\begin{aligned} x_{ip}^d &= x_{i\overline{p}+1} \\ j^d &= \overline{j}. \end{aligned}$$

Using these properties we can dualise any discrete function f given by a lattice expression in discrete variables.

3.1 Dualisation of positive discrete functions

From equations (13) and (14) of the preceding section it follows that:

$$c_{v,j}^d = a_{\overline{v},\overline{j}} \ . \tag{18}$$

Therefore, the dual of the positive function f given by:

$$f = \bigvee_{v,j} j.c_{v,j} \text{ equals } f^d = \bigwedge_{v,j} \overline{j} \vee a_{\overline{v},\overline{j}}. \tag{19}$$

Thus the dual of a positive function can be computed by exchanging $\wedge$ and $\vee$, replacing constants by their complements and discrete variables x_{ip} by their duals $x_{i\overline{p}+1}$.

Example 3 Consider the ternary function of $f : \{0,1,2\}^3 \mapsto \{0,1,2\}$ defined by:

$$\begin{array}{l} f(x) = \text{ if } x \geq 212 \text{ then } 2 \\ \qquad\quad \text{else if } x \geq 002 \text{ then } 1 \\ \qquad\qquad \text{else } 0. \end{array} \quad \text{Then} \quad \begin{aligned} f(x) &= 2.x_{12}x_{21}x_{32} \vee 1.x_{32} \\ f^d(x) &= (x_{11} \vee x_{22} \vee x_{31})(1 \vee x_{31}) \\ &= 1.x_{11} \vee 1.x_{22} \vee x_{31}. \ \square \end{aligned}$$

The minlist representation of f^d is:

$$\begin{array}{l} f^d(x) = \text{ if } x \geq 001 \text{ then } 2 \\ \qquad\quad \text{else if } x \geq 020, 100 \text{ then } 1 \\ \qquad\qquad \text{else } 0. \end{array}$$

By dualisation of the DNF of f^d we can also derive the CNF of f:

$$f(x) = (1 \vee x_{12})(1 \vee x_{21})x_{32}. \tag{20}$$

The three disjunctions in (20) are the prime implicates of f. The following theorem is a generalisation of a well-known theorem in Boolean function theory.

Theorem 1 *Let f be a positive discrete function. Then:* $\min T_j(f^d) = \{x \in X |\ \overline{x} \in \max T_{\overline{j}}(f)\}$. *Dually,* $\max T_i(f^d) = \{x \in X |\ \overline{x} \in \min T_{\overline{i}}(f)\},\ j \in (m],\ i \in [m)$.

Proof The theorem is a consequence of the following equivalence:

$$x \in \min T_j(f^d) \ \Leftrightarrow\ \overline{x} \in \max T_{\overline{j}}(f). \tag{21}$$

To prove (21), let $x \in \min T_j(f^d)$. Then $f^d(x) = j$, implying $f(\overline{x}) = \overline{j}$. Therefore, $\overline{x} \in T_{\overline{j}}(f)$. Suppose $y > \overline{x}$, so $\overline{y} < x$. Then, since f is positive and x is minimal in $T_j(f^d)$, we conclude $\overline{y} \notin T_j(f^d)$. This is equivalent to $f^d(\overline{y}) \neq j \Leftrightarrow \overline{f}(y) \neq j \Leftrightarrow f(y) \neq \overline{j}$. Thus, if $y > \overline{x}$ then $y \notin T_{\overline{j}}(f)$. This proves the claim that $\overline{x} \in \max T_{\overline{j}}(f^d)$. The converse: $\overline{x} \in \max T_{\overline{j}}(f)$ implies $x \in \min T_j(f^d)$ is proved analogously. □

Theorem 1 can be used to compute $\max T(f)$ from $\min T(f)$.

Example 4 The maxlist representation of the positive function f defined in example 3 follows from the minlist representation of f^d given:

$f(x) =$ if $x \leq 221$ then 0
else if $x \leq 122,202$ then 1
else 2. □

Note, that the maxlist of f is obtained from the minlist of f^d by complementing the minimal vectors as well as the function values, and by reversing the inequalities.

3.2 Interpretation of the dual of a positive function

For positive Boolean functions it is known that $y \in T(f^d)$ if and only if $\forall x \in T(f) : y \not\leq \overline{x}$. This means that y is a transversal of $T(f) :\ y \wedge \overline{x} \neq 0^*$. In particular this means that a prime implicant of f^d has at least one variable in common with every prime implicant of f.

Binary functions

Before we generalise this result for discrete functions, we first note that for binary functions a similar result holds.

Lemma 2 *Let f be a positive binary function. Then:*

$$y \in T(f^d) \Leftrightarrow \forall x \in T(f) :\ y \not\leq \overline{x}. \tag{22}$$

Proof The proof is the same as for Boolean functions, and therefore omitted. □

Note, that the condition $y \not\leq \overline{x}$ is equivalent to:

$$\exists i\ \in (n] \text{ such that } x_i + y_i \geq n_i + 1, \text{ where } n_i + 1 = |X_i|. \tag{23}$$

If we interpret (22) or (23) as y is a 'transversal' of x, then Lemma 2 implies that $\min T(f^d)$ is just the set of minimal transversals of $\min T(f)$. Otherwise stated c_w is

a prime implicant of f^d if and only if c_w is a minimal transversal of the prime implicants of f. Thus, if d_v is any prime implicant of f, then there exist a variable x_{ip} in c_w and a variable x_{iq} in d_v such that $p + q \geq n_i + 1$.

Example 5 Consider the binary function $f : \{0,1,2\}^3 \mapsto \{0,1\}$ defined by
$f = x_{12}x_{21}x_{32} \vee x_{11}x_{22}x_{31}$. Then $f^d = (x_{11} \vee x_{22} \vee x_{31})(x_{12} \vee x_{21} \vee x_{32}) =$
$= x_{12} \vee x_{11} \vee x_{21} \vee x_{22} \vee x_{21} \vee x_{31} \vee x_{32}$. Thus $\min T(f) = \{212, 121\}$ and $\min T(f^d) = \{200, 110, 020, 011, 002\}$ is the set of all minimal 'transversals' of $\min T(f)$. □

Discrete functions

To interpret f^d in the case of a positive discrete function f, we represent f by its binary *monotone* level functions defined in section 2.

Theorem 2 *Let f be a positive discrete function, then*
$\lambda_j(f)^d = \lambda_{\overline{j+1}}(f^d)$ *and* $\lambda_j(f^d) = \lambda_{\overline{j+1}}(f)^d$, $j \in (m]$.

Proof $\lambda_j(f)^d(x) = \overline{\lambda}_j(f)(\overline{x}) = 1 \Leftrightarrow f(\overline{x}) < j \Leftrightarrow f^d(x) > \overline{j} \Leftrightarrow f^d(x) \geq \overline{j} + 1$.
The last inequality is equivalent to $x \in T(\lambda_{\overline{j+1}}(f^d))$. Therefore, $\lambda_j(f)^d = \lambda_{\overline{j+1}}(f^d)$. This also implies $\lambda_j(f^d) = \lambda_{\overline{j+1}}(f)^d$, since $\overline{j} + 1 = \overline{j \Leftrightarrow 1}$. □

We now return to the interpretation of the dual of a positive discrete function f.

Theorem 3 *Let f be a positive discrete function then we have:*
$f^d(y) \geq j \Leftrightarrow y \not\leq x$ *for all x with* $f(x) \geq \overline{j} + 1$.

Proof $f^d(y) \geq j \Leftrightarrow \lambda_j(f^d)(y) = 1 \Leftrightarrow \lambda_{\overline{j+1}}(f)^d(y) = 1$.
Since the last equality implies that y is a transversal of the binary function $\lambda_{\overline{j+1}}(f)$, we conclude $f^d(y) \geq j \Leftrightarrow \forall x : f(x) \geq \overline{j} + 1 \Rightarrow y \not\leq \overline{x}$. □

3.3 Dualisation and class functions

As discussed in section 2, the class functions f_j of a monotone function f are binary functions that indicate for each $x \in X$ whether $f(x) = j$ or not. Since f is binary $T_1(f_j)$ will also be denoted by $T(f_j)$ and $T_0(f_j)$ by $F(f_j)$. However, the class functions are not positive.

The functions f_j can be expressed in terms of the test functions occurring in either a minlist or maxlist representation of f. This can be used to dualise a monotone function f, i.e. to compute $\max T(f)$ from $\min T(f)$ and vice versa.

Example 6 Let f be the positive function $\{0,1,2\}^2 \mapsto \{0,1,2\}$ given by
$\min T_1(f) = \{02, 11, 20\}$ and $\min T_2(f) = \{22\}$. Then $s_1(x) = x \geq 02, 11, 20$ and $s_2(x) = x \geq 22$. Suppose that the maxlist of f is given by: $f(x) = ((t_0(x), 0)); (t_1(x), 1); 2)$.

Then the following holds:

$$f_0 = t_0 = \overline{s}_1\overline{s}_2 \quad (24)$$
$$f_1 = t_1\overline{t}_0 = s_1\overline{s}_2 \quad (25)$$
$$f_2 = \overline{t}_1\overline{t}_2 = s_2 \quad (26)$$

From $s_1(x) = x_{22} \vee x_{11}x_{21} \vee x_{12}$ and $s_2(x) = x_{12}x_{22}$, the class functions f_j can be computed.

Furthermore, we can use f_1 and f_0 to compute the maximal vectors as follows:

$$\max T_0(f) = \max T(f_0) = \max T(\overline{s}_1\overline{s}_2) \quad (27)$$
$$\max T_1(f) = \max T(f_1) = \max T(s_1\overline{s}_2). \quad (28)$$

Since : $\overline{s}_1\overline{s}_2 = (\overline{x}_{22} \wedge (\overline{x}_{11} \vee \overline{x}_{21}) \wedge \overline{x}_{12})(\overline{x}_{12} \vee \overline{x}_{22}) = \overline{x}_{11}\overline{x}_{22} \vee \overline{x}_{12}\overline{x}_{21}$,
and: $s_1\overline{x}_2 = \overline{x}_{12}x_{22} \vee x_{11}\overline{x}_{12}x_{21} \vee x_{11}x_{21}\overline{x}_{22} \vee x_{12}\overline{x}_{22}$,
equations (27) and (28) imply: $\max T_0(f) = \{01, 10\}$ and $\max T_1(f) = \{12, 21\}$. □

In general, if f is a positive function and $s_j, j \in (m]$ are the test functions in the minlist of f, then the following equations hold:

$$f_j = s_j\overline{s}_{j+1}\cdots\overline{s}_{m-1}\overline{s}_m = t_j\overline{t}_{j-1}\cdots\overline{t}_1\overline{t}_0,\ j \in (m \Leftrightarrow 1] \quad (29)$$
$$f_0 = \overline{s}_1\overline{s}_2\cdots\overline{s}_m = t_0 \quad (30)$$
$$f_m = s_m = \overline{t}_{m-1}\overline{t}_{m-2}\cdots\overline{t}_0. \quad (31)$$

Using these equations we can compute $\max T_i(f) = \max T(f_i),\ i \in [m)$, given the functions s_j. Similarly, we can compute $\min T_j(f) = \min T(f_j), j \in (m]$, given the functions t_i. Since this enables us to switch between $\min T(f)$ and $\max T(f)$, these equations can be used to compute the dual of a positive function.

The minimal DNFs of class functions

Since class functions of positive discrete functions are not monotone, they do not have a unique DNF consisting of prime implicants. It is known [21] that a prime implicant of a class function f_j is an indicator function of an interval:

$$c(x) = \begin{cases} 1 \text{ if } x \in [v, w] \\ 0 \text{ otherwise} \end{cases}, \text{ where } v \in \min T_j(f),\ w \in \max T_j \text{ and } v \leq w. \quad (32)$$

Störmer [21] discusses an elaborate method to find all minimal DNFs of a class function. Since this is in fact a set covering problem, we briefly show here how to solve this problem by dualising an appropriate positive Boolean function.

Example 7 We consider the class function of the positive function of 2 variables discussed in the preceding example, see table 2. The prime implicants of f_1 are the indicator functions $c_1, \cdots, c_4$ of respectively the intervals: [02,12], [11,12], [11,21] and [21,22].

2	1	1	2
1	0	1	1
0	0	0	1
	0	1	2

Table 2: The function f

$$f_1 = c_1 \vee c_2 \vee c_4$$
$$f_1 = c_1 \vee c_3 \vee c_4.$$

Figure 1: The minimal DNFs of f_1

For each prime implicant c_i we introduce a Boolean variable x_i. For each element $v \in \min T(f_1)$ we form a conjunction m_v of these variables, such that $x_i \in \mathrm{Var}(m_v) \Leftrightarrow v \in T(c_i)$. Furthermore, the Boolean function g_j is defined by:

$$g_j = \bigwedge_v \{\ m_v \mid v \in \min T(f_j)\}. \tag{33}$$

In this example $g_1 = x_1 \vee x_2x_3 \vee x_4$. The prime implicants of g_1 actually denote that the minimal vectors 02, 11 and 20 are true vectors of c_1, c_2 and c_3, c_4. To find the minimal transversals of the sets $\{c_1\}$, $\{c_2,\ c_3\}$ and $\{c_4\}$, we dualise the function g_1:

$$g^d = x_1x_2x_4 \vee x_1x_3x_4. \tag{34}$$

From (34) we conclude that the class function f_1 admits exactly the two minimal DNFs given in figure 1. □

3.4 Dual comparable discrete functions

Two discrete functions f and g are called *mutually dual comparable*, if one of the following conditions hold: $f \leq g^d$, $f^d \leq g$ or $f^d = g$. The functions f and g are called respectively mutually dual minor, mutually dual major or mutually dual. In particular the function f is called *dual minor, dual major* or *self dual* if $f \leq f^d, f^d \leq f$ or $f^d = f$.

Example 8 Consider the function $f : \{0,1\}^2 \mapsto \{0,1,2\}$ defined by
{00:1, 01:0, 10:2, 11:1}. Then f is non-monotone and self dual. This can be checked by applying the definition $f^d(x) = \overline{f}(\overline{x})$ directly, or by dualising the following DNF expression for f:

$$f = u\overline{v} \vee 1.uv \vee 1.\overline{u}\,\overline{v}, \tag{35}$$

where the binary variables u and v are defined by:

$$u = x_{11} = \begin{cases} 2 & \text{if } x_1 = 1 \\ 0 & \text{if } x_1 = 0 \end{cases} \quad \text{and } v = x_{21} = \begin{cases} 2 & \text{if } x_2 = 1 \\ 0 & \text{if } x_2 = 0 \end{cases} \tag{36}$$

Since u and v are binary variables we have $u^d = u$ and $v^d = v$.
Therefore:

$$f^d = (u \vee \overline{v})(1 \vee u \vee v)(1 \vee \overline{u} \vee \overline{v}) = u\overline{v} \vee 1.u \vee 1.\overline{v} \tag{37}$$

Although, the DNF expression for f^d is different from (35), it is easy to see that $f^d = f$. $\square$

For positive discrete functions the question of dual comparability can be reduced to that of the comparability of positive binary functions.

Theorem 4 *Let f and g be positive discrete functions. Then*
$f \leq g^d \Leftrightarrow \lambda_p(f) \leq \lambda_q(g)^d$ *for all* $p, q:\ p+q = m+1$.

Proof $f \leq g^d \Leftrightarrow \forall p \in (m] : \lambda_p(f) \leq \lambda_p(g^d) = \lambda_{\overline{p}+1}(g)^d$. Here we have used lemma 1 and theorem 2. If $q = \overline{p}+1$, then $p+q = m+1$. Therefore: $f \leq g^d \Leftrightarrow \lambda_p(f) \leq \lambda_q(g)^d$. $\square$

Similar results hold for mutual dual major and mutual dual functions. In particular, a positive function f is self dual iff $\lambda_p(f) = \lambda_q(f)^d,\ \ \forall p, q \in (m]:\ p+q = m+1$.

3.5 Dualisation of pseudo-Boolean functions

A pseudo-Boolean function F is a function $F : X \mapsto \mathbb{R}$, where X is the hypercube $\{0,1\}^n$. Since X is finite, we can also view F as a discrete function: $F : X \mapsto Y$, where $Y = \{\alpha_0, \alpha_1, \ldots, \alpha_m\},\ \alpha_i \in \mathbb{R}$, consists of all different function values of F in increasing order. As a consequence we can also define concepts such as the complement and the dual of a pseudo-Boolean function. The complement of a pseudo-Boolean function F is defined by: $\overline{F}(x) = \overline{\alpha}_j = \alpha_{\overline{j}} = \alpha_{m-j}$, whenever $F(x) = \alpha_j$. As noticed before, without loss of generality we may replace $Y = \{\alpha_0, \alpha_1, \cdots, \alpha_m\}$ by $Y = \{0, 1, 2, \cdots, m\}$. For example, let f denote the associated discrete function: $f(x) = j \Leftrightarrow F(x) = \alpha_j$. Then $f^d(x) = p \Leftrightarrow F^d(x) = \alpha_p$. Therefore, $f^d = f \Leftrightarrow F^d = F$.

Example 9 Let $F : \{0,1\}^2 \mapsto \mathbb{R}$ be the pseudo-Boolean function defined by:
$F(x_1, x_2) = 3x_1 \Leftrightarrow 4x_2 + x_1x_2$. Then F has three distinct function values:
$\alpha_0 = \Leftrightarrow 4,\ \alpha_1 = 0$ and $\alpha_2 = 3$, see table 3.

x_1	x_2	F	F^d	f	f^d
0	0	0	0	1	1
0	1	-4	-4	0	0
1	0	3	3	2	2
1	1	0	0	1	1

Table 3: A self dual pseudo-Boolean function

The associated discrete function f, see table 3, is the same as in example 8. We have already noticed that $f^d = f$. Therefore F is also self dual. The dual of F can also be computed directly by applying the definition $F^d(x) = \overline{F}(\overline{x})$. This shows again that F is self dual. $\square$

4 Monotone extensions of partially defined discrete functions

Discrete functions frequently encountered in machine learning and datamining are only partially known. A *partially defined discrete function* (pdDf) is a function:

$$f : D \mapsto Y, \text{ where } D \subseteq X. \tag{38}$$

We assume that a pdDf f is given by a set of pairs $x : y$, where $x \in D$ and $y \in Y = [m+1)$. Although pdDfs are often used in practical applications, the theory of pdDfs is only developed in the case of pdBfs (partially defined Boolean functions). Here we discuss monotone pdDfs, i.e. functions that are monotone on D. If the function $\hat{f} : X \mapsto Y$, agrees with f on D: $\hat{f}(x) = f(x),\ x \in D$, then $\hat{f}$ is called an *extension* of the pdDf f. The set of all extensions is a distributive lattice: for, if f_1 and f_2 are extensions of the pdDf f, then $f_1 \wedge f_2$ and $f_1 \vee f_2$ are also extensions of f. The same holds for the set of all monotone extensions. The lattice of all *monotone* extensions of a pdDf f will be denoted here by $\mathcal{E}(f)$. It is easy to see that $\mathcal{E}(f)$ is universally bounded: it has a greatest and a smallest element.

Definition 1 *Let f be a monotone pdDf. Then the functions $f_{\min}$ and $f_{\max}$ are defined as follows:*

$$f_{\min}(x) = \begin{cases} \max\{f(y) :\ y \in D \cap {\downarrow}x\} & \text{if } x \in {\uparrow}D \\ 0 & \text{otherwise} \end{cases} \tag{39}$$

$$f_{\max}(x) = \begin{cases} \min\{f(y) :\ y \in D \cup {\uparrow}x\} & \text{if } x \in {\downarrow}D \\ m & \text{otherwise} \end{cases} \tag{40}$$

Lemma 3 *Let f be a monotone pdDf. Then*

a) $f_{\min},\ f_{\max}\ \in \mathcal{E}(f)$.

b) $\forall \hat{f} \in \mathcal{E}(f) :\quad f_{\min} \leq \hat{f} \leq f_{\max}$.

Proof The proof can be found in [19]. □

Since $\mathcal{E}(f)$ is a distributive lattice, the minimal and maximal monotone extension of f can also be described by the following expressions:

$$f_{\max} = \bigvee\{\ \hat{f} \mid \hat{f}\ \in\ \mathcal{E}(f)\} \text{ and } f_{\min} = \bigwedge\{\ \hat{f} \mid \hat{f}\ \in\ \mathcal{E}(f)\}. \tag{41}$$

According to the previous lemma $f_{\min}$ and $f_{\max}$ are respectively the minimal and maximal monotone extension of f. Decision lists of these extensions can be directly constructed from f as follows. Let $D_j := D \cap T_j(f)$, then $\min T_j(f_{\min}) = \min D_j$ and $\max T_j(f_{\max}) = \max D_j$.

Example 10 Consider the pdDf $f = \{001 : 0,\ 002 : 1,\ 112 : 1,\ 202 : 1,\ 212 : 2\}$. Then $\min D_1(f) = \{002\}$ and $\min D_2(f) = \{212\}$. Similarly, $\max D_0(f) = \{001\}$ and $\max D_1(f) = \{112, 202\}$. Therefore, the minlist of $f_{\min}$ and the *max*list of $f_{\max}$ are given by respectively:

$$f_{\min}(x) = \begin{array}{l}\text{if } x \geq 212 \text{ then } 2\\ \quad\text{else if } x \geq 002 \text{ then } 1\\ \quad\quad\text{else } 0,\end{array} \qquad f_{\max}(x) = \begin{array}{l}\text{if } x \leq 001 \text{ then } 0\\ \quad\text{else if } x \leq 112,202 \text{ then } 1\\ \quad\quad\text{else } 2.\end{array}$$ □

Note, that the pdDf f in this example is also discussed in section 3: example 3. To compare $f_{\min}$ and $f_{\max}$, it is convenient to use the same representation for these functions. This can be done by dualisation in order to switch between the minlist and maxlist representation. This yields:

$$f_{\min}(x) = \begin{array}{l}\text{if } x \leq 221 \text{ then } 0\\ \quad\text{else if } x \leq 122,202 \text{ then } 1\\ \quad\quad\text{else } 2,\end{array} \qquad f_{\max}(x) = \begin{array}{l}\text{if } x \geq 210 \text{ then } 2\\ \quad\text{else if } x \geq 100,002,010 \text{ then } 1\\ \quad\quad\text{else } 0.\end{array}$$

Using the minlist representations of $f_{\min}$ and $f_{\max}$ it follows that e.g. the function g defined by:

$$g(x) = \begin{array}{l}\text{if } x \geq 210 \text{ then } 2\\ \quad\text{else if } x \geq 002 \text{ then } 1\\ \quad\quad\text{else } 0,\end{array}$$

is a monotone extension of f satisfying: $f_{\min} < g < f_{\max}$. The advantage of comparing decision list or DNF representations of $f_{\min}$ and $f_{\max}$ is, that we can search for extensions with a more compact representation. This is similar to the minimum description length principle used in machine learning to find hypotheses(extensions) with greater generalisation capabilities. Another possible advantage is that the minimal and maximal extension can guide the construction of monotone extensions by other learning methods, such as monotone decision decision trees [19, 18]. Finally, we remark that the method OLM discussed by Ben-David in [4, 5], can be viewed as an algorithm for constructing $\min T(f_{\min})$, where f is a pdDf, extended with a filter to handle inconsistent data.

5 Identification of positive discrete functions

Since a monotone function is determined by its minimal or maximal vectors, the identification problem can be stated as follows, where the input consists of an oracle for a monotone discrete function f.

Problem IDENTIFICATION

Input: An oracle for a positive discrete function f.
Output: $\min T(f)$ and $\max T(f)$.

The oracle for f answers queries about the function values of input vectors given to it. Given a vector $x \in X$, the oracle returns its answer (i.e. $f(x)$) in time $O(s)$, where $s = \log |Y|$. The input length of the oracle is defined as $O(nr)$, where n is the number of variables and $r = \log \max(|X_i|)$.

In the special case that f is a Boolean function, the oracle is called a membership oracle. In computational learning theory [1, 2], identification of a given Boolean function f by asking membership queries to an oracle whether $f(x) = 0$ or 1 holds for some input vector x, is also called exact learning of a Boolean theory f by membership queries only.

However, note that here by identification we mean to determine *both* the set of minimal vectors $\min T(f)$ as well as the sets of maximal vectors $\max T(f)$. Although $\max T(f)$ can be computed from $\min T(f)$ in principle, we consider identifying of both the set of $\min T(f)$ and $\max T(f)$ as essential. During the process of identification which we discuss here partial knowledge of both $\min T(f)$ and $\max T(f)$ becomes available, even in the case we identify the sets $\min T(f)$ and $\max T(f)$ separately. Therefore, maintaining both these sets is natural.

The problem of identification, and the problem of computing $\max T(f)$ from $\min T(f)$ are related to the problem of dualisation of f. The complexity of this problem, discussed in the next section, is still a well-known open problem, see [6, 14]. The problem of identification and dualisation of a positive Boolean function and the relation to many other problems is extensively studied in [6]. It appears that these problems for positive discrete functions can be reduced to solving m identification (dualisation) problems for positive binary functions. However, the problems of dualisation and identification of positive binary functions are not essentially different from those of Boolean functions. Thus, most of the results in [6, 14] for Boolean functions can be generalised to binary functions. Since this generalisation step is more or less straightforward we will not discuss the details of this step in this paper. In the next subsection we present a simple algorithm for the identification of positive discrete functions. This algorithm is polynomial(quadratic) for so-called *stable* functions.

5.1 Identification of positive stable functions

Suppose MIN and MAX denote the partial knowledge of $\min T(f)$ and $\max T(f)$ already available:

$$MIN \subseteq \min T(f) \text{ and } MAX \subseteq \max T(f). \tag{42}$$

Then we can approximate f by two functions g and h defined by:

$$\min T(g) = MIN \text{ and } \max T(h) = MAX. \tag{43}$$

Lemma 4 *Suppose f is a positive discrete function. Let g and h be positive functions defined by (43). Then $g \leq f \leq h$.*

Proof Trivial. □

From the preceding lemma it follows that if MIN increases, then the function g will also increase. Moreover, if MAX increases, then the function h will decrease. Therefore, if the size of MIN or MAX increases, then g or h approaches to f.

Definition 2 *Let f be a positive discrete function. Then f is called stable iff for all pairs of functions g and h defined by subsets of respectively minimal and maximal vectors of f (see equation (43)), the following inequality hold:*

$$|\max T(g)| + |\min T(h)| \leq |\min T(f)| + |\max T(f)|. \tag{44}$$

It is not yet known how large the subclass of stable discrete functions is. In particular we have not found positive Boolean functions that are not stable. Since stable Boolean

functions can be dualised in quadratic total time, it would be interesting to know the size of the class of stable functions compared to the number of positive functions. Furthermore, it would be interesting to know whether the following conjecture is true or not:

Conjecture The class of positive regular Boolean functions is a (proper) subclass of the class of stable functions.

The following theorem implies that to find new maximal vectors of f not contained in $\max T(h)$, it is sufficient to compute the minimal vectors of h (e.g. by dualising h incrementally). This step, which can be done in polynomial time for stable functions will be discussed later in this section.

Theorem 5 *Let $u \in \min T_j(h)$. Then either $f(u) = h(u) = j$ and $u \in \min T_j(f)$ or $f(u) < j$. If $f(u) = i < j$, then there exists a vector $w \geq u$ such that $w \in \max T_i(f) \setminus \max T_i(h)$.*

Proof Since $f \leq h$ and $h(u) = j$, we have $f(u) \leq j$. First suppose that $f(u) = j$. Then $u \in T_j(f)$. Let $v \in \min T_j(f)$ and $v \leq u$. Then $h(v) \geq f(v) = j$. Since h is positive we also have $h(v) \leq h(u) = j$. Therefore, $h(v) = j$, implying $v \in T_j(h)$. From $u \in \min T_j(h)$ and $v \leq u$ we conclude $v = u$. This proves that $u \in \min T_j(f)$.

Finally, in the case that $f(u) = i < j$, we claim that $\forall w \in \max T_i(h)$ the following holds: $u \not\leq w$. For, if $u \leq w$ for some maximal vector of $T_i(h)$, then $j = h(u) \leq h(w) = i$. This contradicts the assumption $i < j$. From $u \in T_i(f)$ and $u \not\leq w$ for all $w \in \max T_i(h)$, we conclude: $\exists w' \geq u, w' \in \max T_i(f)$ and $w' \notin \max T_i(h)$. □

The above theorem implies that if all minimal vectors of h are also minimal vectors of f, then we have identified $f : f = h$. Otherwise, we can find a new maximal vector of f not contained in $\max T(h)$. Dually we can formulate this theorem for the function g.

Theorem 6 *Let $u \in \max T_j(g)$. Then either $g(u) = f(u) = j$ and $u \in \max T_j(f)$ or $f(u) > j$. If $f(u) = i > j$, then there exists a vector $v \leq u$ such that $v \in \min T_i(f) \setminus \min T_i(g)$.*

So if $g \neq f$, we can find a new minimal vector of f not contained in $\min T(g)$ by computing the maximal vectors of g. Computing a minimal true vector or a maximal false vector w of a positive Boolean function from an unknown vector u can be done in $O(n)$ time. For discrete functions we have the following generalisation of an algorithm in [6].

Algorithm MAXIMAL

Input: An incomparable set $M_j \leq \max T_j(f)$, a vector $u \in T_j(f) \setminus M_j$, and an oracle for f, where f is a positive discrete function.

Output: A maximal vector $w \in \max T_j(f) \setminus M_j$.
1. $w := u; j := f(u)$
2. for $i = 1$ to n do
   while $w_i < m_i$ and $f(w) = j$ do $w_i := w_i + 1$;
   if $f(w) > j$ then $w_i := w_i + 1$
3. Output w.

Minimal vectors can be computed analogously. This algorithm is called MINIMAL. Both algorithms use $O(n)$ queries in step 2 before outputting the maximal (minimal) vector w.

Before discussing our identification algorithm we present an algorithm to identify a positive function f given an oracle for f, by identifying $\min T(f)$. The correctness of this algorithm, called IdMIN, follows from Theorem 6. We can also identify f by computing only $\max T(f)$ by a similar algorithm IdMAX.

Algorithm IdMIN

Input: An oracle for a stable discrete function f
Output: $\min T(f)$
1. Initialisation
   $v_0 :=$ MINIMAL(all-one);
   Define the positive function g by: $\min T_0(g) = \{v_0\}$.
2. Update $\max T(g)$.
3. Test if there exists a vector $w \in \max T_j(g)$ for some j such that $f(w) > j$.
   If not, then output $\min T(f)$ and halt. Otherwise goto 4.
4. If $f(w) = i$ and $i > j$, then compute a new minimal vector v in $T_i(f)$:
   $v :=$ MINIMAL(w), and update g by $\min T(g) := \min T(g) \cup \{v\}$. Goto 2.

In step 2 $\max T(g)$ is updated by incrementally updating the dual of g as follows. At each iteration g is updated by adding a new minimal vector v, or equivalently by adding a new prime implicant c_v to g. Thus $g := g_{old} \vee c_v$. Therefore, $g^d := g^d_{old} \wedge c^d_v$. Since f is stable, this step can be carried out in time $O(|f| + |f^d|)$. Here $|f|$ is the size of f: the total length of all minimal vectors of f.

In step 4 of algorithm IdMIN a new minimal vector is computed. In general there are many vectors v for which $g(v) \neq f(v)$. Therefore, in step 4 we can compute in general more than one minimal vector. Furthermore, as shown in Theorem 6, if $g(v) = f(v)$, then v is a maximal vector of f. Therefore, during the identification of $\min T(f)$, we also get more and more information about $\max T(f)$. This information could be used to identify simultaneously the maximal vectors of f using IdMAX. Thus running IdMIN and IdMAX in 'parallel' such that all available knowledge can be used by both of them can reduce the number of iterations substantially.

From our discussion it follows that the running time of algorithm IdMIN is $O(|\min T(f)|m_f)$, where m_f is defined as: $|f|+|f^d|$. Here we assume that in each iteration at most one minimal vector of f is computed. Thus, the running time of the algorithm IdMIN is at most $O(m_f^2)$.

Example 11 As an example we consider the identification of the Boolean function $f = x_1x_2 \vee x_2x_3x_4 \vee x_1x_4$, where we assume of course that we only have a membership oracle of f at our disposal. The dual of f is $f^d = x_1x_2 \vee x_1x_3 \vee x_1x_4 \vee x_2x_4$. Therefore, we have to identify the following sets, where $T(f)(F(f))$ denotes $T_1(f)(T_0(f))$.

$$\min T(f) = \{1100, 0111, 1001\} \text{ and } \max F(f) = \{0011, 0101, 0110, 1010\} \tag{45}$$

After the initialisation steps of IdMIN and IdMAX we have

$$\min T(g) = \{1100\} \text{ and } \max F(h) = \{1010\} \tag{46}$$

By dualising the functions g and h defined in (46) we get:

$$\max F(g) = \{0111, 1011\} \text{ and } \min T(h) = \{0100, 0001\} \tag{47}$$

Using the oracle for f, (45) and (47) imply that $\min T(h)$ does not contain minimal vectors of $T(f)$. Therefore, using MAXIMAL we find two new maximal vectors of $F(f)$: 0110 and 0101. Similarly, $\max F(g)$ does not contain maximal vectors of f. Using MINIMAL, we obtain two new minimal vectors of $T(f)$: 0111 and 1001. The new vectors are added to the sets in (46). Thus, after one iteration we have:

$$\min T(g) = \{1100, 0111, 1001\} \text{ and } \max T(h) = \{1010, 0110, 0101\} \tag{48}$$

Note, that if $\min T(h)$ did contain minimal vectors of $T(f)$, then we would have added these to $\min T(g)$ also. Similarly, maximal vectors of $F(f)$ found in $\max F(g)$ would have been added to $\max T(h)$. Thus, by intertwining the algorithms IdMIN and IdMAX, we obtain maximal information in each iteration.

Although it seems that we have identified $\min T(f)$ in one iteration, since $\min T(g) = \min T(f)$, we need one more iteration, because our algorithm only can halt in step 3 of IdMIN and/or IdMAX. A second iteration of IdMIN yields:

$$\max F(g) = \{0011, 0101, 0110, 1010\} \tag{49}$$

Using the oracle we find that all the vectors in (49) belong to $F(f)$, and therefore to $\max F(f)$. Therefore, the algorithm halts in step 3 of IdMIN and outputs the sets given in (45). The general procedure for identifying a positive discrete function f can now be formulated as follows.

Algorithm IDENTIFY

Input: An oracle for a positive (stable) discrete function f.
Output: $\min T(f)$ and $\max T(f)$.
1. Initialise the positive functions g and h by step 1 of respectively IdMIN and IdMAX.
2. Update $\max T(g)$ and $\min T(h)$ (as in IdMIN and IdMAX);
   MAX:= $\{u \in \max T(g) \mid g(u) = h(u)\}$;
   MIN:= $\{u \in \min T(h) \mid f(u) = h(u)\}$.
3. If MAX $\neq \max T(g)$ then
   $\min T(g) := \min T(g) \vee \{v \mid v = \text{MINIMAL}(u); u \in \max T(g) \setminus \text{MAX}\} \vee \text{MIN}$,
   goto 2. Otherwise $f = g$: $\min T(f) := \min(T(g)$; $\max T(f) := \max T(g)$; halt.
4. If MIN $\neq \min T(h)$ then
   $\max T(h) := \max T(h) \vee \{w \mid w = \text{MAXIMAL}(u); u \in \min T(h) \setminus \text{MIN}\} \vee \text{MAX}$,
   goto 2. Otherwise $f = h$: $\min T(f) := \max(T(f)$; $\max T(f) := \max T(h)$; halt. □

If we assume that at each iteration at most one maximal and one minimal vector of f is found, then the running time of the algorithm IDENTIFY is $O(m_f^2)$. This follows from our analysis of the running time of the algorithm IdMIN and the fact that each step in the algorithm can be carried out in time $O(m_f)$. Note, that algorithm Identify does not require that f is a stable positive function. However, if f is not stable, then of course there is no guarantee that the running time is quadratic in m_f. For the class of positive discrete functions identification can be done in (quasi-polynomial) time $O(m_f Q(m_f))$

where $Q(m_f) = m_f^{O(\log m_f)}$. This is can be done by representing f by its binary level functions and by generalising a number of results proved by Fredman and Khachiyan in [14], and by Bioch and Ibaraki in [6].

6 Conclusions

Multi-attribute decision making is essentially based on (partially) defined discrete functions. Therefore, the theory of partially defined discrete functions (pdDFs) is an important tool for the analysis of multi-attribute data sets.

In this paper in particular monotone discrete functions are studied for ordinal classification and exact learning of a monotone discrete function. However, since very little is known about the class of monotone discrete functions, this class remains an interesting topic for further research. Decision lists appear to be a natural and useful representation of monotone discrete functions. The relationship between decision lists and the disjunctive (conjunctive) normal form is discussed. We have shown how to dualise a decision list and how they can be used to construct monotone extensions of monotone pdDFs.

We have extended the notion of the dual of a function, well-known in Boolean function theory, to discrete functions. It appears that dualisation plays an important rôle in the study of monotone discrete functions. By introducing so-called level functions of a discrete function, it can be shown that the study of monotone discrete functions can be mainly reduced to that of monotone binary functions.

We have argued that pseudo-Boolean functions can be viewed as discrete functions and it is shown how to dualise pseudo-Boolean functions.

The (related) problems of the complexity of dualisation and identification of monotone discrete functions can be reduced to those of monotone binary functions. The complexity of these problems is still not known even for Boolean functions. However, [6, 14], these problems are unlikely to be NP-hard for Boolean functions, and therefore also for monotone discrete functions.

We have introduced the class of monotone stable functions, and presented a polynomial algorithm for the identfication of stable functions. The class of stable functions is an interesting topic for future research.

Acknowledgement

The author thanks Toshihide Ibaraki of Kyoto University and Peter Hammer of Rutgers University for some fruitful discussions on this and related subjects.

References

[1] Angluin, D. (1988). Queries and concept learning. *Mach. Learning 2*, 319-342.

[2] Anthony, M. and N. Biggs (1992). *Computational Learning Theory*, Cambridge University Press, Cambridge, UK.

[3] Agrawal, R. and R. Srikant (1994). Fast Algorithms for Mining Associations Rules. In: *Proc of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile.

[4] Ben-David, A. (1992). Automatic generation of symbolic multiattribute ordinal knowledge-based DSSs: methodology and applications. In: *Decision Sciences*, vol. 23, 1357-1372.

[5] Ben-David A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, vol. 19, 29-43.

[6] Bioch J.C. and T. Ibaraki (1995). Complexity of Identification and Dualisation of Positive Boolean Functions. *Information and Computation*, vol 123, 50-63.

[7] Bioch, J.C. and R. Potharst (1997). Decision Trees for Monotone Classification. In: K. van Marcke and W. Daelmans eds. *Proceedings of the Dutch Artificial Conference on Artificial Intelligence (NAIC97)*, Antwerpen, 361-369.

[8] Boros, E., T. Ibaraki and K. Makino (1995). Error-free and best-fit extensions of partially defined Boolean functions with missing data. RUTCOR Research Report RRR 14-95, Rutgers University.

[9] Boros, E., T. Ibaraki and K. Makino (1996). Extensions of partially defined Boolean functons with missing data. RUTCOR Research Report RRR 06-96, Rutgers University.

[10] Boros, E., P.L. Hammer, T. Ibaraki and A. Kogan (1997). Logical analysis of numerical data. In: T.M. Liebling and D. de Werra, editors, *Mathematical Programming*, North-Holland, vol 79, nos. 1-3 Oct. 1997, 163-191.

[11] Crama, Y., P.L. Hammer and T. Ibaraki (1988). Cause-effect relationships and partially defined Boolean functions. *Annals of Operations Research*, vol. 16, 299-326.

[12] Davio, M., J. Deschamps and A. Thayse (1978). *Discrete and switching functions.* McGraw-Hill.

[13] Fredman, M., and L. Khachiyan (1996). On the Complexity of Dualization of Monotone Disjunctive Normal Forms. *Journal of algorithms* 21, 618-628.

[14] Greco, S., B. Matarazzo and R. Slowinski (1996). Rough Approximation of Preference Relation by Dominance Relations. ICS Research Report 16/96, Warsaw, University of Technology.

[15] Greco, S., B. Matarazzo and R. Slowinski (1997). A new rough set approach to the evaluaton of the bankruptcy risk. Unpublished manuscript.

[16] Hammer, P.L. (1986). Partially defined Boolean functions and cause-effect relationships. In: *Proceedings of the International Conference on Multi-Attribute Decision Making Via OR-Based Expert Systems*, University of Passau, Germany.

[17] Makino, K., T. Suda, K. Yano, and T. Ibaraki (1996). Data analysis by positive decision trees. In: *Proceedings International symposium on cooperative database systems for advanced applications (CODAS)*, Kyoto, 282-289.

[18] Potharst, R., J.C. Bioch and T. Petter (1997). Monotone decision trees. Technical report EUR-FEW-CS-97-07, Dept. of Computer Science, Erasmus University Rotterdam, 47 pp.

[19] Potharst R., J.C. Bioch and R. Dordrecht (1998). Quasi-monotone decision trees. Technical report EUR-FEW-CS-98-02, Dept. of Computer Science, Erasmus University Rotterdam. Technical report EUR-FEW-CS-98-02, Dept. of Computer Science, Erasmus University Rotterdam.

[20] Störmer, H. (1990). Binary Functions and their Applications. *Lecture Notes in Economics and Mathematical Systems*, vol.348. Springer-Verlag Berlin Heidelberg.