THE INCREMENTAL SPLITTING OF INTERVALS ALGORITHM FOR THE DESIGN OF BINARY IMAGE OPERATORS

N. S. T. Hirata[*], J. Barrera, R. Terada
Department of Computer Science
Institute of Mathematics and Statistics – University of São Paulo
Rua do Matão, 1010 – São Paulo, Brazil – 05508-900
nina@ime.usp.br, jb@ime.usp.br, rt@ime.usp.br

Ed. R. Dougherty
Department of Electrical Engineering - Texas A&M University
College Station - 77843-3128 - TX - United States
edward@ee.tamu.edu

Abstract This paper discusses the design of binary image operators from training data and its relation to Boolean function learning. An extended version of the incremental splitting of intervals (ISI) algorithm for Boolean function learning and some improvements and heuristics to reduce its processing time are proposed. Some examples illustrate the application of the algorithm.

Keywords: Binary image operator, Boolean function minimization, learning.

1. Introduction

Boolean functions characterize a large class of binary image operators. Designing these operators from pairs of observed/ideal training images can be viewed as designing or learning Boolean functions. From the input-output data collected from the sample images, one can build part of the truth-table of a Boolean function. However, in practice, the quantity of entries in this table is relatively small. In order to fill the table and also to find a more compact representation for the function, incompletely specified Boolean function minimization algorithms can be applied [2, 3]. This framework is briefly recalled in Section 2.

[*]Partially supported by CNPq - Brazil, proc. 68.0093/01-0 and 300238/01-0.

In Section 3, an extension of the incremental splitting of intervals (ISI) algorithm [2, 3], for Boolean function minimization, is presented. More specifically, the basic operation of splitting an interval by a single element is extended to the operation of splitting an interval by a sub-interval, leading to a more general and efficient algorithm. Some improvements and heuristics that aim to reduce the processing time of the algorithm are also presented.

In Section 4, some application examples are shown and in Section 5 our concluding remarks and steps for future research are presented.

2. Designing Binary Image Operators

A binary image on $E = Z^2$ is a function from E to $\{0,1\}$, or equivalently, a subset of E. A set $S \subseteq E$ represents the binary image $1_S : E \to \{0,1\}$ defined, for all $x \in E$, by $1_S(x) = 1 \iff x \in S$.

Let $W \subset E$, $W = \{w_1, w_2, \ldots, w_n\}$, be a non-empty set called *window*, and $\psi : \{0,1\}^n \to \{0,1\}$ be a Boolean function on n variables. Let $1_S|W_x$, where W_x is the translation of W by x, denote the vector $(1_S(x+w_1), 1_S(x+w_2), \ldots, 1_S(x+w_n))$. The mapping $\Psi : \mathcal{P}(E) \to \mathcal{P}(E)$ defined, $\forall x \in E$ and $\forall S \subseteq E$, as

$$x \in \Psi(S) \iff \psi(1_S|W_x) = 1$$

is a binary image operator characterized by function ψ.

The *kernel* of Ψ is defined as being the set $\mathcal{K}(\Psi) = \{X \in \{0,1\}^n : \psi(X) = 1\}$. The collection of all maximal intervals of the kernel of Ψ corresponds to the *basis* of Ψ, denoted $\mathbf{B}(\Psi)$.

Given an interval $[A,B] \subseteq \{0,1\}^n$, the sup-generating operator is the one characterized by $\lambda_{[A,B]}(X) = 1 \Leftrightarrow A \le X \le B$. Binary image operators have a canonical representation in terms of elements in its basis [3, 1]:

$$1_{\Psi(S)}(x) = max\{\lambda_{[A,B]}(1_S|W_x) : [A,B] \in \mathbf{B}(\Psi)\}.$$

One who is familiar to the notion of kernel and basis of an operator will immediately recognize that these notions correspond, respectively, to the canonical sum of products and minimal sum of products form of a Boolean function. Therefore, estimating the kernel of the operators is equivalent to estimating the ones of the functions, and finding the basis of the operators is equivalent to minimizing Boolean functions.

Given pairs of observed/ideal images, each pair representing an image one wants to process and its respective ideal desired result, the design goal is to find an operator that, when applied to the observed images, generates images that best approximates the respective ideal images. This can be stated as an optimization problem: an operator ψ^* in a family $\mathcal{C}$ is optimal in $\mathcal{C}$ with respect to a given error measure $Er(\cdot)$ if

$$Er(\psi^*) \le Er(\psi),\ \forall \psi \in \mathcal{C}.$$

The error of an operator is related to the process associated to the observed/ideal images. In our case, the observed/ideal images are considered realizations of a jointly stationary process $(\mathbf{S}, \mathbf{I})$. The mean absolute error of ψ, $MAE(\psi)$, at a given position $x \in E$, is defined as

$$MAE\langle \psi, x \rangle = \sum_{(S,I) \in \mathcal{P}(E) \times \mathcal{P}(E)} |1_{\Psi(S)}(x) - 1_I(x)| P(1_S|W_x, 1_I(x))$$

where $P(1_S|W_x, 1_I(x))$ is the joint probability of $S \cap W_x$ and $1_I(x)$.

Due to stationarity, we can omit x, so that

$$MAE\langle \psi \rangle = \sum_{(X,y) \in \{0,1\}^n \times \{0,1\}} |\psi(X) - y| \, P(X, y)$$

where X is the vector $1_S|W_x$ and $y = 1_I(x)$. It can be shown that the MAE-optimal operator is the one characterized by the following Boolean function:

$$\psi^*(X) = \begin{cases} 1, & \text{if } P(X,1) > P(X,0), \\ 0, & \text{if } P(X,1) \leq P(X,0). \end{cases} \tag{1}$$

The design procedure considered here is composed of three basic steps:

1 estimation of the joint probabilities $P(X, y)$ from pairs of sample images;

2 definition of $\psi(X)$ by using the estimated probabilities in place of $P(X, y)$ in equation 1;

3 attribution of a value for ψ for those non-observed shapes (generalization).

Step 3 can be accomplished, for instance, via incompletely specified Boolean function minimization. The resulting intervals characterize product terms of the final Boolean expression and can be directly mapped to the morphological representation in terms of erosions and dilations.

3. The incremental splitting of intervals algorithm

The incremental splitting of intervals (ISI) is a Boolean function minimization algorithm that has been used in the context of designing binary image operators in the last years [3, 2]. This section presents a more general version of the ISI algorithm and some recent improvements [6].

3.1 Basic definitions and concepts

Given $A, B \in \mathcal{P}(W)$, the collection $[A, B] = \{X \in \mathcal{P}(W) : A \subseteq X \subseteq B\}$ is called the interval with extremities A and B.

Given $\mathcal{X} \subseteq \mathcal{P}(W)$, an interval $[A, B] \subseteq \mathcal{X}$ is *maximal* in $\mathcal{X}$ if there is no other interval $[A', B'] \subseteq \mathcal{X}$ such that $[A, B] \subset [A', B']$. The set of all maximal intervals of a collection of elements $\mathcal{X} \subseteq \mathcal{P}(W)$ is denoted $\mathbf{M}(\mathcal{X})$.

Proposition 3.1 (Splitting rule) *Let $[A, B]$ and $[C, D]$ be two intervals of $\mathcal{P}(W)$ such that $[A, B] \cap [C, D] \neq \emptyset$. Then*

$$\mathbf{M}([A, B] \setminus [C, D]) = \left\{[A, B \cap \{c\}^c] : c \in C \cap A^c\right\} \cup \left\{[A \cup \{d\}, B] : d \in D^c \cap B\right\}.$$

Here, $\mathbf{M}([A,B] \setminus [C,D])$ means the maximal intervals in the collection $[A,B] \setminus [C,D]$. As we can see, it can be written as a union of two collections of intervals, the ones with left extremity A and the ones with right extremity B. This result generalizes the splitting by a single element (or trivial interval) that has been previously presented [2, 3].

Example 3.1 *Let* $[A,B] = [000, 111]$ *(also denoted* XXX*) and* $[C,D] = [001, 011]$ *(also denoted* 0X1*). The elements of* $\{0,1\}^3$ *appear as circles in the Hasse diagram representation in Fig. 1. The elements of the interval appear as the filled circles while the others appear as non-filled circles. There are two maximal intervals contained in* XXX$\setminus$0X1*:* $\{[000, 110]$ *and* $[100, 111]\}$ *(or* XX0 *and* 1XX*).*

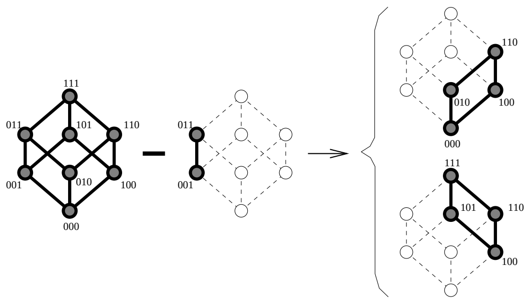

Figure 1. The maximal intervals contained in XXX $\setminus$ 0X1 are XX0 and 1XX.

3.2 The algorithm

The splitting rule presented in the previous section is the core of ISI algorithm. Starting from the n-cube, ISI successively removes elements of $\psi\langle 0\rangle = \{X \in \{0,1\}^n : \psi(X) = 0\}$ by splitting the intervals by these elements. After the splitting step, when all elements of $\psi\langle 0\rangle$ have been removed, the remaining elements are represented by a collection of maximal intervals contained in it. At the end of the splitting process, the remaining elements are exactly the elements in $\psi\langle 1\rangle = \{X \in \{0,1\}^n : \psi(X) = 1\}$, represented by the collection of maximal intervals contained in it. A key point here is to note that the maximal intervals contained in $\psi\langle 1\rangle$, i.e., $\mathbf{M}(\psi\langle 1\rangle)$, form the basis of the corresponding operator Ψ.

Example 3.2 *Consider the minimization of* $\psi(x_1,x_2,x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3$*. The elements of* $\psi\langle 0\rangle = \{111, 011, 010\}$ *are given by the collection* $\mathbb{I}_{\psi\langle 0\rangle} = \{$X11, 01X$\}$*. Figure 2 shows the maximal intervals computation process by the ISI algorithm. ISI starts with the* 3*-cube* XXX*. The first splitting step eliminates elements of interval* X11*. The resulting intervals are* $\{$XX0, X0X$\}$*. The second splitting step eliminates elements of interval* 01X*. Since this interval intercepts* XX0*, it is split and generates intervals* 1X0 *and* X00*. But since* X00 $\subseteq$ X0X*, it can be eliminated. Thus, after the splitting phase finishes, the remaining intervals are* $\{$X0X, 1X0$\}$*.*

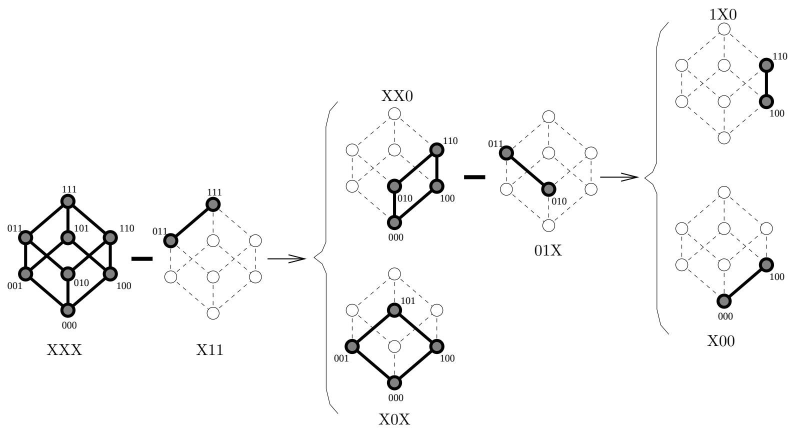

Figure 2. ISI algorithm dynamics.

The ISI algorithm consists of two basic steps: (i) computation of the maximal intervals in the kernel (i.e., basis), and (ii) selection of a sub-collection of the intervals that is enough to represent the kernel (i.e., a sub-basis). A description of (ii) can be found, for instance, in [5].

In many cases, particularly for large values of n, Boolean functions are not completely specified. In our case, the value of the function is not known for those shapes that have not been observed during sampling. These elements are called *don't cares* and will be denoted as $\psi\langle * \rangle$.

If ISI is applied as described above to the set $\psi\langle 1\rangle \cup \psi\langle 0\rangle \cup \psi\langle * \rangle$, then after all elements of $\psi\langle 0\rangle$ are extracted from the initial n-cube, the resulting intervals are all maximal intervals contained in $\psi\langle 1\rangle \cup \psi\langle * \rangle$. However, there is no need to cover elements in $\psi\langle * \rangle$ because they can be equally mapped to 0 or 1. If during the splitting process, an interval that does not contain any element of $\psi\langle 1\rangle$ is generated, it can be discarded because it will not be needed in step (ii).

Figure 3 shows a pseudo-code of the ISI algorithm that takes into account these observations. Lines 2-14 correspond to step (i) and line 15 to step (ii). For each interval $[C, D]$ representing the zeros of the function (line 2), ISI separates the current set of intervals into two sets: those that does and does not intercept $[C, D]$ (lines 3-4). Only those that intercept $[C, D]$ need to be split (lines 6-12). An interval that is split is destroyed, and from those that are newly generated, only those that are not contained in some already existing interval and those that contain at least one element of $\psi\langle 1\rangle$ are kept (line 13). After removing all elements of $\psi\langle 0\rangle$, the algorithm proceeds to step (ii) (line 15).

3.3 Heuristics and analysis

Step (i) of ISI algorithm starts with one interval and then splits it into smaller intervals, which in their turn may be further split, and so on. Therefore, as the number of iterations increases so does the number of newly generated intervals. For a large value of n, the number of intervals generated may become

```
1.  $\mathbb{I} \longleftarrow \{[\emptyset, W]\}$ ;
2.  For each $[C, D] \in \mathbb{I}_{\psi\langle 0\rangle}$ do {
3.      $\mathbb{I}_P \longleftarrow \{[A, B] \in \mathbb{I} : [A, B] \cap [C, D] = \emptyset\}$ ;
4.      $\mathbb{I}_{tmp} \longleftarrow \{[A, B] \in \mathbb{I} : [A, B] \cap [C, D] \neq \emptyset\}$ ;
5.      $\mathbb{I}_{New} \longleftarrow \emptyset$;
6.      For each $[A, B] \in \mathbb{I}_{tmp}$ do {
7.          $\mathbb{I}_{split} \longleftarrow \mathbf{M}([A, B] \setminus [C, D])$ ;
8.          For each $[A', B'] \in \mathbb{I}_{split}$ {
9.              if there is no $[E, F] \in \mathbb{I}_P$ such that $[A', B'] \subseteq [E, F]$
9'.             and if $[A', B']$ covers at least one element of $\psi\langle 1\rangle$
10.                 then $\mathbb{I}_{New} \longleftarrow \mathbb{I}_{New} \cup \{[A', B']\}$ ;
11.         }
12.     }
13.     $\mathbb{I} \longleftarrow \mathbb{I}_P \cup \mathbb{I}_{New}$ ;
14. }
15. $\mathbb{I} = min\_cover(\psi\langle 1\rangle, \mathbb{I})$ ;
16. Return $\mathbb{I}$.
```

Figure 3. ISI algorithm.

computationally prohibitive even with the improvement mentioned above. A heuristic to keep a relatively small number of intervals at any iteration of the algorithm consists in computing a minimum cover regularly after a fixed number of iterations of the splitting phase. This regular number of iterations is called a *period* and denoted k. If $k = 1$, then the number of intervals after each iteration is no more than $|\psi\langle 1\rangle|$. To implement this heuristic, in the algorithm of Fig. 3, the minimum cover (line 15) must be computed between lines 13 and 14, whenever the iteration is a multiple of k. This heuristic may increase the number of resulting intervals because some intervals that would be part of an optimal solution may be thrown away during the extraction process.

In practice, the gain in processing time due to the heuristic is far more attractive than the slight increase in the number of resulting intervals. Moreover, the increase in the number of resulting intervals does not significantly affect the precision of the designed operator. The following tables illustrate two experimental cases (MAE has been measured on a independent set of test data)

4. Application examples

An image operator $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ is *anti-extensive* if $\Psi(S) \subseteq S$, for all $S \subseteq E$. In terms of its characteristic Boolean function, anti-extensiveness means that if $o \notin X$ then $\psi(X) = 0$ (o denotes the reference point of the window). In the case of anti-extensive operators, any element of $\psi\langle 1\rangle$ is necessarily an element of the interval $[o, W]$. Therefore, the ISI algorithm can be modified to start with this interval instead of the interval $[\emptyset, W]$. This restriction reduces the possible operators to those whose kernel is a subset of the interval $[o, W]$.

Table 1. ISI for different periods ($n = 15$, $|\psi\langle 1\rangle| = 9110$ and $|\psi\langle 0\rangle| = 14538$).

k	$\lvert\mathbb{I}\rvert$	Time	MAE
1	1472	13807.91	0.009290
10	1385	1402.54	0.009295
25	1391	558.61	0.009296
50	1348	338.98	0.009307
100	1334	250.99	0.009317
300*	1327	203.30	0.009308
800	1321	253.70	0.009318
1500	1310	337.03	0.009321
3000	1345	522.89	0.009324
5000	1405	873.22	0.009319
8000	1470	1370.79	0.009307
12000	1412	1437.90	0.009316
14538	1523	3498.07	0.009313

Table 2. ISI for different periods ($n = 25$, $|\psi\langle 1\rangle| = 1340$ and $|\psi\langle 0\rangle| = 8126$).

k	$\lvert\mathbb{I}\rvert$	Time	MAE
1	423	382.99	0.015918
10*	350	51.57	0.013547
25	328	84.70	0.012615
50	318	299.26	0.012484
100	320	1147.23	0.012732
300	317	21954.26	0.012785
800	344	204746.53	0.013227

The design technique discussed here is initially applied to the observed/ideal pairs of images (S, I). Once a first operator Ψ_1 is designed, a second one Ψ_2 to enhance the result of Ψ_1 can be similarly designed from pairs $(\Psi_1(S), I)$. This can be successively iterated, for instance, until no more improvements are achieved [8]. $\Psi_1(S)$ is the result of iteration 1, $\Psi_2(\Psi_1(S))$ is the result of iteration 2 and so on.

Figure 4 shows the result obtained for the recognition of adders and multipliers in a functional diagram, using a 9×7 window. A total of 6 pairs of similar images has been used for learning the operator.

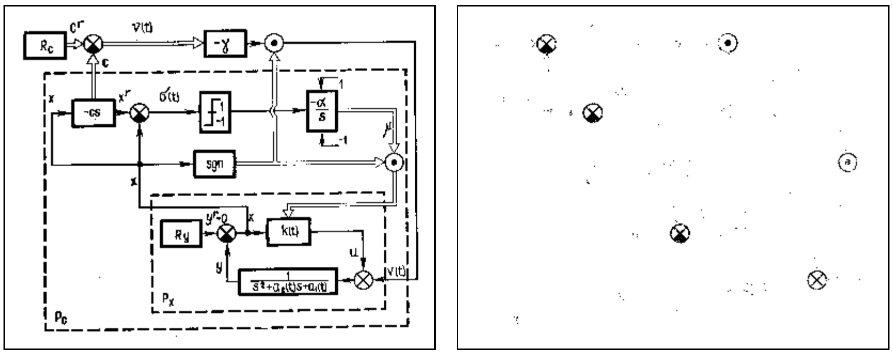

Figure 4. Extraction of objects from a diagram.

Figure 5 shows the result of a two-iteration operator, both designed on 7×5 window, for segmenting texture from map images. A total of three pairs of training images have been used.

This technique has also been applied in character recognition problems [4, 7].

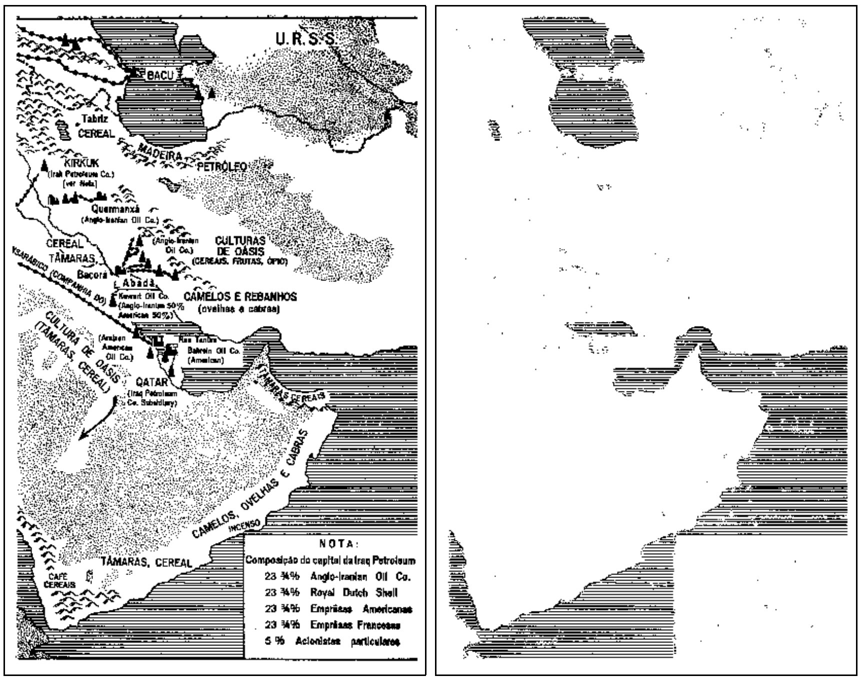

Figure 5. Texture segmentation: test image and result of a two-iteration operator.

5. Concluding remarks

This paper recalled the design of binary image operators from training examples given by observed/ideal pairs of images. The problem was posed under the perspective of Boolean function minimization. An extended version of the ISI algorithm and some improvements and heuristics have been considered. Some examples show the results obtained.

For large windows, usually the amount of data is not enough for obtaining a good precision. A possible approach to overcome this difficulty is to reduce the space of operators by considering a sub-family of the whole family. An interesting question is how these constraints will affect the Boolean function learning problem.

References

[1] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM J. Applied Mathematics*, 51(6):1782–1798, December 1991.

[2] J. Barrera, E. R. Dougherty, and N. S. Tomita. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.

[3] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.

[4] M. Brun, J. Barrera, N. S. T. Hirata, N. W. Trepode, D. Dantas, and R. Terada. Multi-resolution Classification Trees in OCR Design. In D. L. Borges and S.-T. Wu, editors, *Proceedings of Sibgrapi 2001*, pages 59–66, Florianopolis, Brasil, October 2001. IEEE.

[5] F. J. Hill and G. R. Peterson. *Introduction to Switching Theory and Logical Design*. John Wiley, 3rd edition, 1981.

[6] N. S. T. Hirata, J. Barrera, and E. R. Dougherty. Boolean Function Minimization by Incremental Splitting of Intervals. *submitted*, 2001.

[7] N. S. T. Hirata, J. Barrera, and R. Terada. Text Segmentation by Automatically Designed Morphological Operators. In *Proc. of SIBGRAPI'2000*, pages 284–291, Gramado, Brazil, October 2000.

[8] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Iterative Design of Morphological Binary Image Operators. *Optical Engineering*, 39(12):3106–3123, December 2000.