Two-stage Binary Image Operator Design: an Approach Based on Interaction Information

Carlos S. Santos, Nina S. T. Hirata and Roberto Hirata Junior
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo
Rua do Matão, 1010 – 05508-090 São Paulo, Brazil
(csantos, nina, hirata)@ime.usp.br

Abstract

We address the problem of binary image operator design over large windows by breaking it into two phases. Firstly, we design several operators over small sub-windows of the main window. The outputs of these first level operators are then combined into a global operator. We devise a heuristic, motivated by Information Theory, for breaking the main window into sub-windows. Preliminary results show that the proposed scheme improves operator performance over single-level operators.

1. Introduction

The design of translation invariant and locally defined binary image operators is equivalent to the design of Boolean functions whose variables are given by the window which establishes the locality property of the operator. An optimal mean absolute error (MAE) operator can be obtained if the joint distribution between images to be processed and respective ideal output images are known. Since in practice this probability is not known, a usual approach is to estimate it from sample pairs and use the estimated probabilities to design an approximately optimal operator [1]. However, for large windows several shortcomings arise: (1) estimation is not accurate, (2) many of the patterns are not even observed in the training sample, and (3) computational time and memory space required is large.

To deal with these difficulties, a two-level design approach based on stacked generalization, a pattern classification approach proposed by Wolpert in [9], has been recently proposed [3]. More specifically, given a large window, several sub-windows, whose union equals the given window, are considered and one operator is designed for each sub-window. This step generates the first level operators. In the sequel, outcomes of these first level operators are combined to form a new pattern that is posteriously used to train a second level operator. This composition results in a two-level operator that indirectly makes use of all data observed through window W, although individual operators of the first level observe only part of it. The advantages of this approach are: (1) the MAE of the estimated composition is better than the MAE of the operator estimated over the whole window and (2) the overall computation time required is smaller than the time required for designing directly on the large window.

So far, sub-windows of the first level operators have been chosen empirically, both with regard to their shapes and quantity, in a trial and error basis [3]. In this paper, we address the problem of selecting appropriate sub-windows of the large window. The proposed approach is based on information theory concepts, particularly on *interaction information* [7]. Information theory has been used before in designing binary morphological operators. In [6], information on the training set is maximized to find a minimal window. In [8], mean conditional entropy is used as a criterion to choose a multiresolution pyramid of windows. In this paper, we present a method based on interaction information (a generalization of mutual information) to select a set of sub-windows of a larger window. This is done by a series of transformations on the variables that define the larger window in a way that the information is preserved but the interaction between the variables is sequentially reduced. At the end, each new variable will lead to a desired sub-window.

This paper is organized as follows. In Section 2, we present a brief review on morphological image operators, their equivalence to Boolean functions and the usual design procedure from training data as well as the two-level training proposed in [3]. In Section 3, we review some main concepts from Information Theory. In Section 4, we present and show that appropriate transformations of binary variables have the property of preserving the information of the ini-

tial variables. By choosing an adequate sequence of binary variable transformations, we are able to generate a new set of variables (related to the initial ones) that preserves information but reduces interaction between pairs of variables. In Section 5 we show how such transformations can be used to select a set of sub-windows to be used in the first level operators. In Section 6 we present some experimental results and in Section 7 we present some discussions and the conclusions of our work.

2. Binary Image Operator Design

Binary images defined on $E = \mathbb{Z}^2$ can be represented as subsets $S \subseteq E$. Let $\mathcal{P}(E)$ be the collection of all subsets of E. The translation of a set $S \in \mathcal{P}(E)$ by a vector $z \in E$ is denoted S_z. Binary image operators are mappings from $\mathcal{P}(E)$ to $\mathcal{P}(E)$. Given an image operator $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$, we say that Ψ is translation-invariant (t.i.) if and only if $[\Psi(S)]_z = \Psi(S_z)$ for any $z \in E$ and $S \in \mathcal{P}(E)$. Given a non-empty subset $W \subseteq E$, $W = \{w_1, w_2, \ldots, w_n\}$, we say that Ψ is locally defined (l.d.) within W if $x \in \Psi(S) \iff x \in \Psi(S \cap W_x)$. If Ψ is both t.i. and l.d., then it can be locally characterized by a function of the form $\psi : \{0,1\}^n \rightarrow \{0,1\}$, on n binary variables $X_1, X_2, \ldots, X_n$, as follows $z \in \Psi(S) \iff \psi(S_{-z} \cap W) = 1$, where $\psi(S_{-z} \cap W)$ means $X_i = 1 \iff w_i \in S_{-z} \cap W$.

2.1. Operator optimality

We assume that there exists a jointly stationary process $(\mathbf{S}, \mathbf{I})$, whose realizations are observed images that we would like to process and their corresponding ideal images (i.e., the images desired as the result of the processing), respectively. The process $\mathbf{S} \cap W_z$ is also a random set, which can be thought as a random vector $\mathbf{X}_z$ by associating each element of $\mathbf{S} \cap W_z$ to a component of $\mathbf{X}_z$. Due to stationarity, we drop z from $\mathbf{X}_z$ and consider that $\mathbf{X}$ is a random vector whose realizations correspond to the patterns observed in $\mathbf{S}$ through W at any location. Similarly, the value of a given pixel z in $\mathbf{I}$ can be thought as a realization of a random variable Y_z. By the same reason, we drop z from Y_z. We denote $P(\mathbf{X}, Y)$ the joint distribution of $(\mathbf{X}, Y)$.

The MAE of an image operator Ψ, characterized by ψ, with respect to process $(\mathbf{S}, \mathbf{I})$ is the expected value $MAE\langle\Psi\rangle = E[|\psi(\mathbf{X}) - Y|]$. It is known that the MAE optimal operator Ψ is the one characterized by the function given by, for any realization $\mathbf{x}$ of $\mathbf{X}$, $\psi(\mathbf{x}) = 0$, if $P(\mathbf{x}, 0) > P(\mathbf{x}, 1)$, $\psi(\mathbf{x}) = 1$, if $P(\mathbf{x}, 1) > P(\mathbf{x}, 0)$, $\psi(\mathbf{x}) = 0$ or 1, otherwise.

2.2. Designing from examples

Computation of an optimal MAE operator depends on the joint distribution P. However, this probability is unknown in general. These probabilities can be estimated from sample pairs (S_i, I_i), $i = 1, \ldots, m$ of input-output images, leading to the following design procedure.

1. Slide W on each input image S_i and at each location record the pattern $\mathbf{x}$ and the respective value y in I_i at the same location. This yields and estimate of $\hat{P}(\mathbf{X}, Y)$ of $P(\mathbf{X}, Y)$.
2. For each observed pattern $\mathbf{x}$, make $\hat{\psi}(\mathbf{x}) = 1 \iff \hat{P}(\mathbf{x}, 1) > \hat{P}(\mathbf{x}, 0)$ and $\hat{\psi}(\mathbf{x}) = 0$ otherwise. This yields an incompletely specified function $\hat{\psi}$ (since, in general, not all possible patterns are observed in step 1).
3. Using the pairs $(\mathbf{x}, \hat{\psi}(\mathbf{x}))$ as training data, apply any generalization algorithm. This yields a completely specified function $\hat{\psi}$. We use Boolean function minimization as the learning algorithm [4].

For the two-level design, let W denote the whole large window and let $W_1, W_2, \ldots, W_k \subset W$ be the sub-windows of the first-level operators. The procedure described above is applied for each of the sub-windows, resulting in k first level operators that we denote $\psi_i^{(1)}$, $i = 1, 2, \ldots, k$. These values form a new random vector, $(\psi_1, \psi_2, \ldots, \psi_k)$, that will be used in the same way, i.e., the pairs $(\psi_1, \psi_2, \ldots, \psi_k, \hat{\psi}(\mathbf{x}))$ will be used as training data to the second-level operator.

3. Review on Information Theory Concepts

In this section we review some concepts from Information Theory which allow us to characterize the redundancy among first-level operators and also the information loss caused by decomposing the problem into parts for posterior combination.

3.1. Entropy and Mutual Information

In the framework of Information Theory, *entropy* is the measure of the mean information necessary for determining the value of one random variable. For a discrete random variable X taking values in a set $\mathcal{X}$ and with distribution $P(X)$, the entropy is defined as [2]:

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \,. \tag{1}$$

Throughout this article we will use 2 as the base for the logarithm, in which case the entropy is measured in bits (binary

digits). The concept can be extended for two or more random variables by replacing $P(X)$ in equation (1) with the joint distribution of the variables. For instance, let X be as above and let Y be a discrete random variable taking values in a set $\mathcal{Y}$. The *joint entropy* $H(X,Y)$ is then given by:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x,y)\,. \qquad (2)$$

The *mutual information* between two random variables, which measures how much uncertainty about one variable is resolved by observation of the other [2] is given by

$$\begin{aligned} I(X;Y) &= H(Y)+H(X)-H(X,Y) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}\,. \end{aligned} \qquad (3)$$

Let $\tilde{X}=\xi(X)$ be a random variable obtained by applying some function ξ on X. The *Data Processing Inequality* [2] states that $\tilde{X}$ cannot convey more information about Y than X itself: $I(\tilde{X};Y)\leq I(X;Y)$.

3.2. Interaction Information

Interaction information [7] is a measure of how much two random variables interact to determine the value of a third one. Let Y, X_1 and X_2 be discrete random variables defined on sets $\mathcal{Y}$, $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. Their interaction information $I(Y;X_1;X_2)$ is defined as:

$$I(Y;X_1;X_2) = I(Y;(X_1,X_2)) - I(Y;X_1) - I(Y;X_2) \qquad (4)$$

where $I(Y;(X_1,X_2))$ is the mutual information between Y and the pair (X_1,X_2) considered jointly.

Interaction information is negative when X_1 and X_2 are *redundant* with respect to Y, i.e. when $I(Y;X_1)$ and $I(Y;X_2)$ overlap. Interaction information is positive when new information about Y is gained by the joint observation of (X_1,X_2) compared to the situation when we only observe X_1 and X_2 separately. Equation (4) can also be written as:

$$I(Y;X_1;X_2) = I(X_1;X_2|Y) - I(X_1;X_2) \qquad (5)$$

where the mutual information between X_1 and X_2 conditioned on the observation of Y, $I(X_1;X_2|Y)$, is given by:

$$I(X_1;X_2|Y) = \sum_{x_1}\sum_{x_2}\sum_{y} P(x_1,x_2,y)\log\frac{P(x_1,x_2|y)}{P(x_1|y)P(x_2|y)}\,. \qquad (6)$$

Interaction information is related to the ability of creating statistical models by parts-to-whole decompositions. When $I(Y;X_1;X_2)$ approaches zero, it is justifiable to estimate the relationship between pairs (Y,X_1) and (Y,X_2) separately and combine them in a output prediction afterwards. In case of negative interaction, $I(Y;X_1)$ and $I(Y;X_2)$ will overlap and one must be careful not to overcount the evidence for any class label $y\in\mathcal{Y}$. When $I(Y;X_1;X_2)$ is positive, decomposition of the problem into parts will lead to information loss; ideally one should try to learn the full relationship between Y and the pair (X_1,X_2) instead.

4. Information Preserving Transformations for Binary Variables

In this section we show a procedure to build mappings from one set of binary variables $\mathbf{X}$ into another set $\tilde{\mathbf{X}}$ while preserving the information about an output variable Y. Moreover, such mappings can be built by observation of only low-order marginal distributions, thus avoiding the complicated issue of estimating high dimensional joint distributions.

4.1. Building Mappings Through Distribution Permutations

Consider once again a random vector $\mathbf{X}=(X_1,X_2,\ldots,X_n)\in\{0,1\}^n$. We choose an arbitrary pair of variables X_i,X_j, $1\leq i<j\leq n$. Define the auxiliary random vector $\mathbf{X}'=(X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_{j-1},X_{j+1}\ldots,X_n)$. The joint distribution of $(X_1,X_2,\ldots,X_n)$ can be written as:

$$\begin{aligned} P(\mathbf{X}) &= P(X_1,X_2,\ldots,X_n) \\ &= P(\mathbf{X}'|X_i,X_j)P(X_i,X_j) \end{aligned} \qquad (7)$$

with $P(\mathbf{X}')$ defined accordingly. Let $\mathrm{F}=(\sigma,\rho)$ be a mapping from $\{0,1\}^2$ to $\{0,1\}^2$, where (σ,ρ) defines a permutation of $\{(0,0),(0,1),(1,0),(1,1)\}$. The following relation is then satisfied by a distribution Q:

$$Q(\sigma(r,s),\rho(r,s)) = P(r,s), \qquad (8)$$

for all $r,s\in\{0,1\}$. One example of such a mapping is $(\tilde{X}_i,\tilde{X}_j)=(\overline{X}_i,\overline{X}_j)$, which leads to: $Q(\overline{r},\overline{s})=P(r,s)$ where $\overline{\bullet}$ means the Boolean negation of $\bullet$.

We define now the random vector $\tilde{\mathbf{X}}=(X_1,\ldots,X_{i-1},\tilde{X}_i,X_{i+1},\ldots,X_{j-1},\tilde{X}_j,X_{j+1}\ldots,X_n)$. The distribution $Q(\tilde{\mathbf{X}})$ of the components of $\tilde{\mathbf{X}}$ can be written as:

$$\begin{aligned} Q(\tilde{\mathbf{X}}) &= Q(X_1,\ldots\tilde{X}_i,\ldots,\tilde{X}_j,\ldots,X_n) \\ &= Q(\mathbf{X}'|\tilde{X}_i,\tilde{X}_j)Q(\tilde{X}_i,\tilde{X}_j)\,. \end{aligned} \qquad (9)$$

We analyze now the joint entropy $H(\tilde{\mathbf{X}})=H(X_1,\ldots\tilde{X}_i,\ldots,\tilde{X}_j,\ldots,X_n)$:

$$H(\tilde{\mathbf{X}}) = -\sum_{\tilde{\mathbf{x}}} Q(\tilde{\mathbf{x}})\log Q(\tilde{x}_i,\tilde{x}_j)Q(\mathbf{x}'|\tilde{x}_i,\tilde{x}_j)\,. \qquad (10)$$

Using standard properties of the logarithm function it can be shown that:

$$H(\tilde{\mathbf{X}}) = H(\tilde{X}_i, \tilde{X}_j) - \sum_{\tilde{\mathbf{x}}} Q(\tilde{\mathbf{x}}) \log Q(\mathbf{x}'|\tilde{x}_i, \tilde{x}_j)\,. \quad (11)$$

The summation term in equation (11) can be further expanded as (using equation (9)):

$$\sum_{\{\tilde{x}_i, \tilde{x}_j\}} Q(\tilde{x}_i, \tilde{x}_j) \sum_{\mathbf{x}'} Q(\mathbf{x}'|\tilde{x}_i, \tilde{x}_j) \log Q(\mathbf{x}'|\tilde{x}_i, \tilde{x}_j)\,. \quad (12)$$

Since there is a one-to-one correspondence between the events $(\tilde{X}_i = \sigma(r,s), \tilde{X}_j = \rho(r,s))$ and $(X_i = r, X_j = s)$, we can write:

$$Q(\mathbf{x}'|\tilde{X}_i = \sigma(r,s), \tilde{X}_j = \rho(r,s)) = P(\mathbf{x}'|X_i = r, X_j = s)\,. \quad (13)$$

By applying the identities (8) and (13) into (12) it is readily seen that the summation in (12) can be recast as:

$$\sum_{\{x_i, x_j\}} P(x_i, x_j) \sum_{\mathbf{x}'} P(\mathbf{x}'|x_i, x_j) \log P(\mathbf{x}'|x_i, x_j)\,. \quad (14)$$

The term in (14) is equivalent to: $\sum_{\mathbf{x}} P(\mathbf{x}) \log P(\mathbf{x}'|x_i, x_j)$. Recalling expression (2) for the joint entropy, it becomes clear that the effect of the mapping $(\tilde{X}_i, \tilde{X}_j) = \mathrm{F}(X_i, X_j)$ over the calculation of the joint entropy $H(\tilde{X}_i, \tilde{X}_j)$ is only to promote a reordering of the terms in the summation, thus the joint entropy remains unaltered:

$$\begin{aligned} H(\tilde{X}_i, \tilde{X}_j) &= -\sum_r \sum_s Q(\sigma\bullet, \rho\bullet) \log Q(\sigma\bullet, \rho\bullet) \\ &= -\sum_r \sum_s P(r,s) \log P(r,s) = H(X_i, X_j)\,. \end{aligned} \quad (15)$$

where $\bullet = (r,s)$. By analogy with equation (11), a similar expression can be derived for $H(\mathbf{X})$:

$$H(\mathbf{X}) = H(X_i, X_j) - \sum_{\mathbf{x}} P(\mathbf{x}) \log P(\mathbf{x}'|x_i, x_j)\,. \quad (16)$$

The equivalence between (12) and (14) and identity (15) implies:

$$\begin{aligned} H(\mathbf{X}) &= H(X_i, X_j) - \sum_{\mathbf{x}} P(\mathbf{x}) \log P(\mathbf{x}'|x_i, x_j) \\ &= H(\tilde{X}_i, \tilde{X}_j) - \sum_{\tilde{\mathbf{x}}} Q(\tilde{\mathbf{x}}) \log Q(\mathbf{x}'|\tilde{x}_i, \tilde{x}_j) \\ &= H(\tilde{\mathbf{X}})\,. \end{aligned} \quad (17)$$

Equation (17) states that mappings that obey the permutation condition defined in (8) will keep constant the joint entropy of a random vector of binary variables. This result entails that for a random vector $\{Y, X_1, X_2, \ldots, X_n\}$ and a mapping $(\tilde{X}_i, \tilde{X}_j) = \mathrm{F}(X_i, X_j)$, both identities will be observed:

$$\begin{aligned} H(\mathbf{X}) &= H(\tilde{\mathbf{X}}) & (18) \\ H(Y, \mathbf{X}) &= H(Y, \tilde{\mathbf{X}})\,. & (19) \end{aligned}$$

Since the mutual informations $I(Y, \mathbf{X})$ and $I(Y, \tilde{\mathbf{X}})$ are given by:

$$\begin{aligned} I(Y, \mathbf{X}) &= H(Y) + H(\mathbf{X}) - H(Y, \mathbf{X}) & (20) \\ I(Y, \tilde{\mathbf{X}}) &= H(Y) + H(\tilde{\mathbf{X}}) - H(Y, \tilde{\mathbf{X}}) & (21) \end{aligned}$$

the mutual information between the output variable Y and the transformed vector $\tilde{\mathbf{X}}$ will also be preserved.

4.2. Permutation Transformations for Binary Variables

Here we study a class of transformations $(\tilde{X}_i, \tilde{X}_j) = \mathrm{F}(X_i, X_j)$ that possesses the permutation characteristic stated in equation (8) and how they affect the interaction. Our goal is finding transformations that reduce the interaction $I(Y; \tilde{X}_i; \tilde{X}_j)$, so the problem of learning an operator $Y = \psi(\mathbf{X})$ can be framed in a parts-to-whole statistical model. This kind of procedure is motivated by previous work [5] showing that feature construction based on interaction information can improve the performance of the Naive Bayes classifier.

Since we have four different events associated with realizations of (X_i, X_j), namely $\{(0,0), (0,1), (1,0), (1,1)\}$, the class of permutation transforms has $4! = 24$ members.

The first fact we must note is that, as shown above, this class of transformations will keep constant the joint entropy, $H(\tilde{X}_i, \tilde{X}_j) = H(X_i, X_j)$. Therefore, the only way to change the value of the mutual information $I(\tilde{X}_i, \tilde{X}_j) = H(\tilde{X}_i) + H(\tilde{X}_j) - H(\tilde{X}_i, \tilde{X}_j)$ is by changing the sum of marginal entropies $H(\tilde{X}_i)$ and $H(\tilde{X}_j)$ with respect to their counterparts $H(X_i)$ and $H(X_j)$. Some permutations will keep that sum unchanged. For instance: $(\tilde{X}_i, \tilde{X}_j) = (X_j, X_i)$ will lead to $H(\tilde{X}_i) = H(X_j)$ and $H(\tilde{X}_j) = H(X_i)$, obviously making $H(\tilde{X}_i) + H(\tilde{X}_j) = H(X_i) + H(X_j)$. Another type of transform that does not affect marginal entropies is negating either one or both variables. To see that, we analyze the entropy formula for one binary variable:

$$H(X) = -\sum_{x \in \{0,1\}} P(X = x) \log P(X = x)\,. \quad (22)$$

Since $P(X = 0) = P(\overline{X} = 1)$ and $P(X = 1) = P(\overline{X} = 0)$, we have that $H(X) = H(\overline{X})$. From the discussion above we can state that for any pair (A, B) of binary variables there can be defined 8 pairs that correspond

to permutation transforms which preserve the mutual information (note that the first one corresponds to the identity transform):

$$\{(A,B),(\overline{A},B),(A,\overline{B}),(\overline{A},\overline{B}),(B,A),(B,\overline{A}),(\overline{B},A),(\overline{B},\overline{A})\}$$

It follows that the set of 24 permutation transforms can lead to at most $24/8 = 3$ different values of mutual information. From equation (5) we see that $I(Y;\tilde{X}_i;\tilde{X}_j)$ can be expressed as a function of mutual information terms between $\tilde{X}_i$ and $\tilde{X}_j$. Therefore permutation transforms can also lead to at most 3 different values of interaction information. One such value is given by the original interaction $I(Y;X_i;X_j)$. The other two correspond to the mappings $(\tilde{X}_i,\tilde{X}_j) = (X_i, X_i \oplus X_j)$ and $(\tilde{X}_i,\tilde{X}_j) = (X_i \oplus X_j, X_j)$, where $\oplus$ denotes the exclusive OR (XOR) operation. Table 1 illustrates how these mappings relate to permutations of events in $\{0,1\}^2$. The XOR operation changes the marginal distribution of the transformed variable:

$$P(X_i \oplus X_j = 0) = P(X_i = 0, X_j = 0) + P(X_i = 1, X_j = 1)$$
$$P(X_i \oplus X_j = 1) = P(X_i = 0, X_j = 1) + P(X_i = 1, X_j = 0)$$

and consequently the entropy $H(X_i \oplus X_j)$ will also be changed.

X_i, X_j	$X_i, X_i \oplus X_j$	$X_i \oplus X_j, X_j$
$0,0$	$0,0$	$0,0$
$0,1$	$0,1$	$\mathbf{1,1}$
$1,0$	$\mathbf{1,1}$	$1,0$
$1,1$	$\mathbf{1,0}$	$\mathbf{0,1}$

Table 1. The two permutation transforms that change mutual information. Permuted events are shown in bold.

The successive application of two-variable mappings will require the calculation of the XOR function over pairs of already transformed variables. In this situation the XOR function generalizes as the *parity function* (PAR) over a set of binary variables $\{Z_1, Z_2, \ldots, Z_K\}$, $Z_k \in \{0,1\}$:

$$\mathrm{PAR}(Z_1, Z_2, \ldots, Z_K) = \begin{cases} 0 & \text{if } \sum_{k=1}^{K} Z_k \text{ is even,} \\ 1 & \text{if } \sum_{k=1}^{K} Z_k \text{ is odd.} \end{cases} \quad (23)$$

The XOR function is equivalent to PAR for two binary variables. Given two sets of binary variables, $\mathbf{X}_\alpha = \{X_{\alpha(1)}, \ldots, X_{\alpha(U)}\}$ and $\mathbf{X}_\beta = \{X_{\beta(1)}, \ldots, X_{\beta(V)}\}$, $\alpha(u), \beta(v) \in \{1, \ldots, n\}$, we have:

$$\mathrm{PAR}(\mathbf{X}_\alpha) \oplus \mathrm{PAR}(\mathbf{X}_\beta) = \mathrm{PAR}(\mathbf{X}_\alpha \cup \mathbf{X}_\beta \setminus \mathbf{X}_\alpha \cap \mathbf{X}_\beta)\,. \quad (24)$$

Variables in the intersection set $\mathbf{X}_\alpha \cap \mathbf{X}_\beta$ are removed because terms that appear twice (or any even number of times) do not influence parity computation. Expression (24) shows that iterative application of two-variable transformations leads to the formation of clusters of binary variables.

We define now a procedure for finding a transformation $\tilde{\mathbf{X}} = \mathrm{G}(\mathbf{X})$, $\mathrm{G} : \{0,1\}^n \mapsto \{0,1\}^n$ which reduces the average magnitude of interactions $I(Y;\tilde{X}_i;\tilde{X}_j)$, $i,j \in \{1,\ldots,n\}$, $Y \in \{0,1\}$. The inputs to the procedure are:

- training pairs $(\mathbf{X},Y)_1, \ldots, (\mathbf{X},Y)_L$
- number of iterations N

The procedure returns:

- the set of transformed vectors: $\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_L$
- a list of clusters $\mathrm{C}[1], \ldots, \mathrm{C}[n]$

Each cluster $\mathrm{C}[\alpha] = \{X_{\alpha(1)}, \ldots, X_{\alpha(U)}\}$, $\alpha(i) \in \{1,\ldots,n\}$, is related to one component of $\tilde{\mathbf{X}}$ by the formula: $\tilde{X}_\alpha = \mathrm{PAR}(\mathrm{C}[\alpha])$.

As the first step of the procedure we set $\tilde{\mathbf{X}} \leftarrow \mathbf{X}$ and define a list of clusters: $\mathrm{C}[i] \leftarrow \{X_i\}$, $i \in \{1,\ldots,n\}$. At each iteration we choose a pair of indices $i,j \in \{1,\ldots,n\}$, $i \neq j$ and calculate the three possible interaction values:

$$\begin{aligned} I_0 &= I(Y;\tilde{X}_i;\tilde{X}_j) \\ I_i &= I(Y;\tilde{X}_i \oplus \tilde{X}_j;\tilde{X}_j) \\ I_j &= I(Y;\tilde{X}_i;\tilde{X}_i \oplus \tilde{X}_j)\,. \end{aligned}$$

If the minimal interaction corresponds to I_0 we just advance to the next iteration. Otherwise we set: $\lambda = \arg\min_{i,j}(I_i, I_j)$ and apply the corresponding mapping, by updating $\tilde{X}_\lambda$, $\tilde{X}_\lambda \leftarrow \tilde{X}_i \oplus \tilde{X}_j$ and the respective cluster:

$$\mathrm{C}[\lambda] \leftarrow \mathrm{C}[i] \cup \mathrm{C}[j] \setminus \mathrm{C}[i] \cap \mathrm{C}[j]$$

This process is repeated till the specified number of iterations is reached.

For assessing the effectiveness of the proposed method we compare interaction values before and after the transformation. For that we adopted three measures:

1. Maximum interaction: $\max_{i,j} I(Y;\theta_i,\theta_j),\ i \neq j$
2. Minimum interaction: $\min_{i,j} I(Y;\theta_i,\theta_j),\ i \neq j$
3. Average of absolute interaction:

$$\frac{2}{n(n-1)} \sum_{i,j} |I(Y;\theta_i,\theta_j)|,\ i \neq j$$

When we calculate pre-transformation interactions θ_i is replaced with X_i. For post-transformation evaluation θ_i is replaced with $\tilde{X}_i$.

5. Sub-Window Selection Based on Interaction Analysis

Recalling the framework for operator design presented in section 2.2, we devise the following scheme for sub-window selection based on interaction information. A large window W is adopted for generating a sample of input-output pairs $(\mathbf{X}, Y)$. After that we apply the above proposed algorithm for interaction reduction. At this point it might be suggested the use of parity functions as the first level operators in a two-stage operator design scheme. Preliminary results (unreported) with this approach showed that an excessive number of parity functions was needed to achieve an acceptable operator performance. That put too much burden on the second level training phase, because of the number of input variables. Alternatively, we adopted the heuristic of treating each derived cluster as one sub-window. The Data Processing Inequality asserts that each sub-window carries at least the same information about Y as its parity function. Thus, there is no information loss in this process.

Let $\mathrm{PAR}(W_i)$ be the parity calculated over sub-window W_i. We select windows based on the mutual information $I(Y, \mathrm{PAR}(W_i))$, as this quantity is a lower bound for the information contained in the window. Subwindows W_i associated to each cluster are ranked in decreasing order of mutual information $I(Y, \mathrm{PAR}(W_i))$ and the first (i.e. most informative) k windows are selected.

6. Experimental Results

We performed experiments with different sets of images, using the above methodology. All experiments follow a common scheme. Given a set of input-output image pairs, a large rectangular window is defined and the algorithm for finding the interaction reducing transform is applied. The quality of the transform is assessed by comparison between pixel interactions (pre-transform) and parity function interactions (post-transform). In experiments reported below, the indices (i, j) were chosen in a deterministic order and the number of iterations was defined so that each combination (i, j) was used at least five times. This process showed to significantly reduce average interactions in all image sets studied.

Two-stage binary operator design was then performed. For that the resulting clusters were treated as sub-windows, which were ranked in decreasing order of mutual information $I(Y, \mathrm{PAR}(W_i))$. After that one first level operator ψ_i was trained for each sub-window in the sequence $(W_1, W_2, \ldots, W_n)$. A sequence of second-level operators was then trained by adding one first-level operator at each iteration, according to the window order. For instance, the initial second-level operator combines the outputs of the first-level operators trained on windows W_1 and W_2; the i-th second-level operator combines the outputs of the first-level operators trained on windows W_1 through W_{i+1}. The same input-output image pairs used to find the windows were employed in training both the first-level and second-level operators. The assessment of second-level operators was performed with a set of testing images, disjoint with the training set. For further evaluation, we also designed a single-level operator over the large window and compared its test error with the one obtained with the two-stage operators.

6.1. Text Segmentation

Identification of text regions is an important task in document processing and analysis. In this experiment, four input-output image pairs similar to the ones shown in Fig. 2 were used for training and another four pairs were used for testing. A 5×7 window was the large window employed for two-level training. Numerical evaluation of the interaction reduction is summarized in table 2. Some of the resulting windows are shown in Fig. 1.

	Interaction Information (bits)		
	Max	Min	Mean Abs (Std Dev)
Pixels	0.0662	-0.0062	0.0221 (0.0107)
Parity functions	0.0139	-0.0267	0.0021 (0.0032)

Table 2. Values for maximum and minimum interactions and mean value of absolute interaction, text segmentation images.

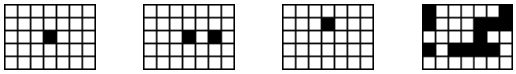

Figure 1. Windows W_1 through W_4 for text segmentation

Figure 2 depicts the behavior of the training and testing errors as more windows are added to the training of the second-level operator, jointly with the test error achieved with a single-level operator trained on the large window. In Fig. 3 we show the result of the best two-level operator for one image in the test set.

6.2. Texture Recognition

Two input-output image pairs similar to the ones shown in Fig 6 were used for training and another three pairs were

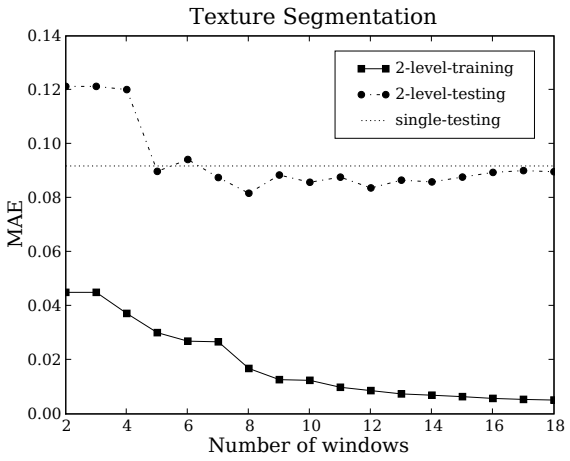

Figure 2. Error in text segmentation.

	Interaction Information (bits)		
	Max	Min	Mean Abs (Std Dev)
Pixels	0.0776	-0.1748	0.0212 (0.0202)
Parity functions	0.0244	-0.0401	0.0056 (0.0047)

Table 3. Pre and post transform interaction values for texture recognition images.

used for testing. A 7×7 window was the large window employed for two-level training. Evaluation of interactions is presented in Table 3. Some of the resulting windows are shown in Fig. 4. Figure 5 depicts the behavior of the training and testing errors as more windows are added to the training of the second-level operator, jointly with the test error achieved with a single-level operator trained on the large window. Figure 6 shows the result of the best two-level operator for one image in the test set.

6.3. Summary of Results

The proposed algorithm for reducing interactions showed to be effective, since in all cases there has been significant reduction in average magnitude of interactions. In Table 4 we compare, for each image set, the best two-level operator with the single-level operator trained on the large window. We observe that the best two-level operator always outperforms the single-level one, which means that the proposed approach for window selection leads to efficient use of the information contained in the large window while taking advantage of the improved statistical precision provided by estimation over a smaller set of variables.

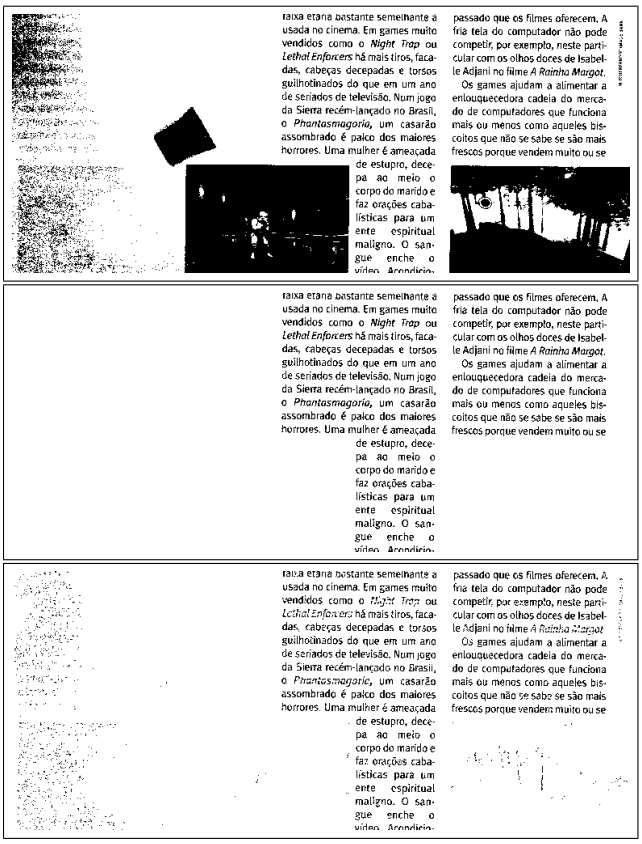

Figure 3. Result of text segmentation. Test, ideal and best two-level operator result.

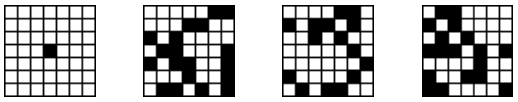

Figure 4. Windows W_1 through W_4 for texture recognition

7. Concluding Remarks

We have presented a method for finding a transformation of binary variables that reduces interactions between them. The method showed to be effective in fulfilling this goal. Based on this method, a heuristic was proposed to find sub-windows used in two-stage design of binary image operators. The operators thus designed outperformed the corresponding single-level operators trained over the full window. Future work will address the issue of selecting the number of windows used in second-level operator training. Another possible improvement is using the values of inter-

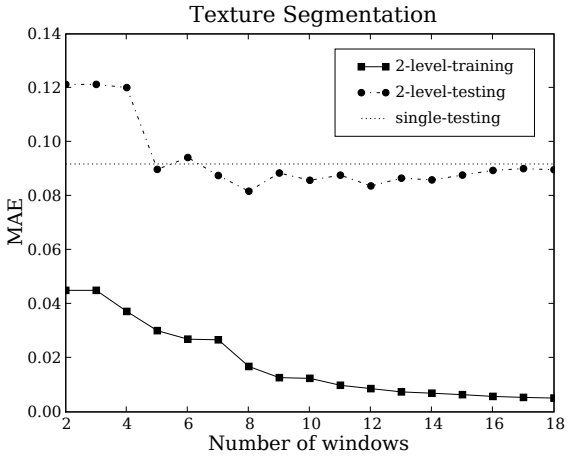

Figure 5. Error in texture recognition.

	# win	Min 2-level MAE	Single-level MAE
Text	13	0.0326	0.0382
Texture	8	0.0817	0.0918

Table 4. Minimum MAE for 2-level operators and MAE for single-level operator trained on large window.

actions obtained after the transformation for guiding window selection. Although the interaction reduction transformation was used here in the context of binary image operator design, the method could be generally used for learning Boolean functions.

Acknowledgements

C. S. Santos acknowledges support from FAPESP (grant 05/04614-7) and from CNPq. N. S. T. Hirata acknowledges support from FAPESP (grant 04/11586-7) and from CNPq (grant 312482/2006-0). R. Hirata Jr. acknowledges support from CNPq. The authors would like to thank the reviewers for their valuable suggestions.

References

[1] 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.

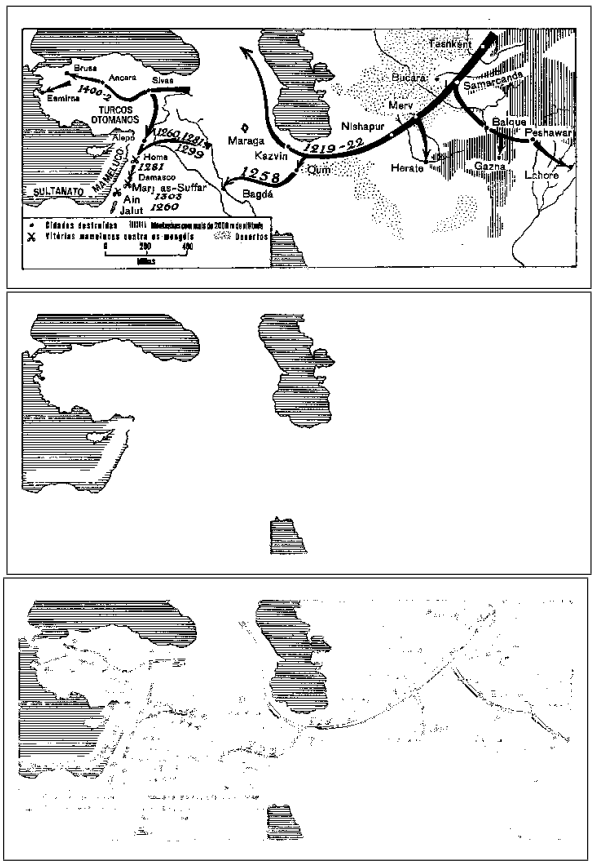

Figure 6. Result of texture recognition. Test, ideal and best two-level operator result.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley series in telecommunications. John Wiley and Sons, 1991.

[3] N. S. T. Hirata. Binary image operator design based on stacked generalization. In *Proceedings of the XVIII SIBGRAPI*, pages 63–70, 2005.

[4] N. S. T. Hirata, J. Barrera, R. Terada, and E. R. Dougherty. The Incremental Splitting of Intervals Algorithm for the Design of Binary Image Operators. In *Proceedings of the 6th ISMM*, pages 219–228, 2002.

[5] A. Jakulin and I. Bratko. Analyzing attribute dependencies. In *7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003)*, 2003.

[6] D. C. Martins Jr., R. M. Cesar Jr., and J. Barrera. W-operator window design by minimization of mean conditional entropy. *Pattern Anal. Appl.*, 9(2):139–153, 2006.

[7] W. J. McGill. Multivariate information transmission. *Psychometrika*, 19:97–116, 1954.

[8] D. A. Vaquero, J. Barrera, and R. Hirata Jr. A maximum-likelihood approach for multiresolution W-operator design. In *Proceedings of the XVIII SIBGRAPI*, pages 71–78, 2005.

[9] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.