Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC

Nina Sumiko Tomita

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE MESTRE
EM
MATEMÁTICA APLICADA

Área de Concentração : **Ciência da Computação**
Orientador : **Prof. Dr. Junior Barrera**

- São Paulo, Fevereiro de 1996 -

Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e apresentada por Nina Sumiko Tomita e aprovada
pela Comissão Julgadora.

São Paulo, 19 de Abril de 1996

Banca Examinadora :

- Prof. Dr. Junior Barrera (orientador) - IME-USP
- Prof. Dr. Routo Terada - IME-USP
- Prof. Dr. Júlio da Motta Singer - IME-USP

aos meus pais

Agradecimentos

Ao Prof. Dr. **Junior Barrera** pela orientação, apoio, incentivo e confiança durante a realização deste trabalho,

À Prof. Dra. **Yoshiko Wakabayashi** pelas orientações e apoio iniciais e nos momentos necessários,

Aos Profs. Drs. **Routo Terada** e **Flávio S. C. da Silva** pela interessante oportunidade de trabalharmos em equipe,

Aos amigos **Roberto Hirata Jr.** e **Francisco de Assis Zampirolli** pela amizade e pela companhia no estudo e nos trabalhos em equipe,

Ao **CNPq** e à **Olivetti do Brasil** pelo apoio financeiro recebido durante a elaboração deste trabalho,

A todos que direta ou indiretamente colaboraram desde o início do mestrado,

E aos meus pais que sempre confiaram e apoiaram meus estudos,

o meu **muito obrigada.**

Resumo

A Teoria de Morfologia Matemática (MM), restrita ao domínio do reticulado Booleano das partes de um conjunto, pode ser utilizada para modelar transformações de imagens binárias.

Os operadores da MM podem ser encarados como frases de uma linguagem formal, denominada Linguagem Morfológica, cujo vocabulário consiste dos operadores elementares da MM (erosão e dilatação) e das operações de interseção, união e complementação de conjuntos.

Estas frases, quando implementadas em máquinas especiais denominadas Máquinas Morfológicas, definem programas que podem ser utilizados para resolver problemas de Processamento de Imagens.

Entretanto, a programação de máquinas morfológicas requer experiência em processamento de imagens e conhecimentos teóricos profundos em MM.

Este trabalho apresenta uma metodologia para programação automática de máquinas morfológicas binárias baseada em processos de aprendizado PAC. Os processos de aprendizado são aqueles nos quais um sistema "aprende" um conceito desconhecido a partir de exemplos que o caracterizam.

Em nosso contexto, os conceitos são os operadores morfológicos e os exemplos que os caracterizam são pares de imagens representando a imagem anterior e posterior a uma determinada transformação desejada.

Um sistema computacional foi implementado com base na metodologia estudada e utilizado para aprender operadores que resolvem vários problemas de Processamento de Imagens.

Este trabalho apresenta também algumas contribuições originais como : a inserção do problema de programação automática de máquinas morfológicas dentro do contexto de aprendizado computacional e a introdução de um novo algoritmo de aprendizado que pode também ser utilizado para a minimização de funções Booleanas.

Abstract

The theory of Mathematical Morphology (MM), restricted to the domain of Boolean complete lattice of subsets, can be used to model mappings between binary images.

The operators of MM can be viewed as expressions of a formal language, denominated Morphological Language, whose vocabulary is composed by the elementary operators (erosion and dilation), and by the set operations (intersection, union and complementation).

These expressions, when implemented on special machines, called Morphological Machines, define programs that can be used to solve Image Analysis problems.

However, the design of these programs is not an elementary task in most cases.

This thesis presents a methodology for automatic programming of binary morphological machines based on PAC learning processes. The learning processes are those in which a system "learns" an unknown concept from examples.

In our context, the concepts are the morphological operators and the examples are pairs of input-output images representing a desired transformation.

A computational system based on the proposed methodology has been implemented and used to learn operators in order to solve several kinds of real Image Analysisproblems.

This thesis also presents some original contributions such as : the insertion of automatic programming of morphological machines in the context of machine learning and the introduction of a new learning algorithm which can be also used to minimize Boolean functions.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Processamento de Imagens

As imagens encontram-se presentes em vários locais sob diversas formas tais como nas fotografias, nos vídeos das TVs, nas telas de cinema, em quadros de museus, em esboços de cavernas, na memória humana ou até mesmo em sonhos. Como objeto de estudo, elas podem ser observadas e analisadas sob os mais variados pontos de vista, desde o artístico até o filosófico.

Iniciamos este texto focalizando nossa atenção para um desses aspectos em especial : a utilização de imagens na resolução de problemas do dia-a-dia. Mostramos, nos seguintes exemplos, algumas atividades nas quais as imagens são utilizadas para a resolução de algum problema real.

- Na área médica, as imagens obtidas por raio-X, tomografia ou ressonância magnética podem ser utilizadas para detecção de fraturas ou lesões e ajudam um médico a emitir um diagnóstico mais preciso.

- As imagens obtidas por satélites são utilizadas para estimação do volume da safra agrícola de uma determinada região, ou ainda, para detecção de áreas de desmatamento em meio às florestas. Elas podem também ser utilizadas para previsão de tempo, análise de vegetações e estudo da topografia da superfície terrestre.

- Em aplicações militares, as imagens são utilizadas, por exemplo, para reconhecimento de alvos tais como instalações militares, portos, aeroportos, movimentaçao das tropas, etc.

- Em biologia e bioquímica, utilizam-se imagens para análise de amostras. São comuns as tarefas de classificação de células e outros elementos com características préestabelecidas a partir de suas imagens. A análise, classificação e identificação de DNA também pode ser realizada a partir de imagens.

- A análise de materiais tais como grãos, metais e rochas é útil para medir a sua resistência ou avaliar o seu grau de pureza. Muitas destas análises são realizadas a partir das imagens de amostras destes materiais.

Em todas as atividades citadas acima, existe um processo de aquisição de imagens através de algum mecanismo de captura (câmeras fotográficas, câmera de vídeo, sensores, lentes, etc). Estas imagens adquiridas podem ser representadas através de um conjunto adequado de dados e posteriormente visualizados, por exemplo, em uma tela de vídeo. Neste texto, a palavra **imagem** é utilizada tanto para referir-se a este conjunto de dados como para a visualização do mesmo. Qualquer operação de manipulação, tranformação ou extração de informações das imagens é denominado genericamente de **processamento de imagem**.

Apesar da importância das imagens para a resolução de vários problemas, a enorme quantidade de dados a serem processados, ou a má qualidade das imagens, ou a própria dificuldade inerente à tarefa que se deseja realizar tornam o processamento manual de imagens uma tarefa difícil.

Com o desenvolvimento da tecnologia, principalmente dos computadores, as pesquisas na área de processamento sistemático e automático de imagens ganharam impulso a partir da década de sessenta . A área de pesquisa denominada **Processamento Digital de Imagens** estuda técnicas de processamento de imagens por computador.

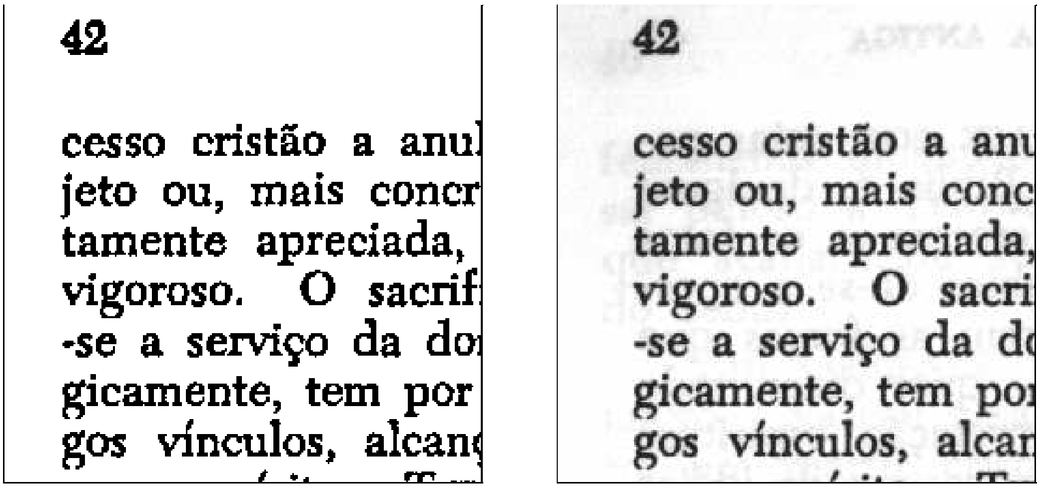
Para serem processadas em computador, as imagens devem estar em formato digital (i.e. discreto). As imagens capturadas em formato analógico (isto é, contínuo), devem ser transformadas para o formato digital. Esta transformação envolve um processo de amostragem e quantização (para maiores informações, consulte por exemplo [GW92]).

As imagens digitais podem ser binárias (caracterizadas por apenas duas cores, preto e branco, que indicam, respectivamente, o objeto e o fundo, como na figura 1.1a), em níveis de cinza (com determinado número de tons de cinza, como na figura 1.1b) ou coloridas. A imagem da figura 1.1a foi obtida a partir da segmentação[1] da imagem da figura 1.1b.

No início da década de sessenta, a NASA[2] iniciou as primeiras pesquisas nesta área através de um projeto que visava a melhoria da qualidade de imagens obtidas por sondas espaciais.

[1]segmentação refere-se a um processo que separa objetos (i.e., as letras em preto) do fundo (i.e., da parte mais clara da figura). Em particular, uma das técnicas de segmentação mais conhecidas é denominada "thresholding".

[2]NASA - National Aeronautics and Space Administration - é um órgão do governo americano que, entre outras coisas, realiza pesquisas de exploração espacial.



a b

Figura 1.1: Imagem (a) binária e (b) em níveis de cinza.

No final dessa mesma década, a aplicação estendeu-se para a área médica, quando começaram a ser processadas as imagens obtidas por raio-X. Na década de setenta, surgiram aplicações sobre imagens obtidas por satélites, tais como a análise da produção agrícola e sistemas para previsão de tempo.

No final da década de setenta, na área de microbiologia passaram a ser realizadas detecção, contagem e classificação automática de células baseadas em processamento de imagens por computador. Nas décadas de oitenta e noventa, começaram a surgir outras aplicações mais sofisticadas como a tomografia computadorizada e a técnica de ressonância magnética na área médica, e os sistemas baseados em visão computacional para controle de processos de produção na área industrial.

O processamento de imagens por computador, além de agilizar o processamento em si, contribuiu para o desenvolvimento de várias técnicas e para a diversificação das aplicações. Uma boa introdução a estas técnicas e à área de Processamento Digital de Imagens em geral podem ser encontradas em algumas referências como [GW92], [Pra91] e [Ros69].

As operações realizadas em processamento de imagens podem ser classificadas em grupos, de acordo com as suas finalidades. Apresentamos a seguir uma breve descrição da classificação definida por G.A.Baxes em [Bax94] :

- <u>"Enhancement"</u> : Este grupo designa as operações cujo objetivo é a melhoria da qualidade visual das imagens. Em geral, estas operações correspondem a um pré-processamento para facilitar ou permitir uma operação posterior. Alguns exemplos de "enhancement" bastante comuns são o ajuste das cores, brilho e contraste, e realce dos elementos presentes nas imagens.

- Restauração : Também consistem de operações que visam melhorar a qualidade da imagem. Porém, neste caso supõe-se que são conhecidas as origens das degradações que se desejam corrigir.

  Por exemplo, várias imagens obtidas por satélites apresentam ruídos inerentes ao processo de aquisição como as distorções geométricas decorrentes do ângulo das câmeras. A restauração tem como objetivo eliminar estes ruídos.

- Análise de Imagens : são operações que visam a extração de informações tais como medidas, estatísticas e classificação de objetos a partir das imagens.

  Algumas tarefas relacionadas ao processamento de imagens quando consideradas juntamente com o contexto no qual estão inseridas assumem aspectos diferenciados e podem ser encaradas como problemas distintos uns dos outros. Por exemplo, identificar células de diâmetro superior a um dado valor em uma imagem miscroscópica é diferente de detectar objetos com comprimento maior que uma dada medida numa linha de produção de uma fábrica. Da mesma forma, reconhecer letras 'a' em um trecho de texto é diferente de reconhecer objetos com formas arredondadas em meio a objetos quadriculados. Porém, quando consideradas em um contexto mais abstrato, as tarefas do primeiro exemplo podem ser encaradas como um problema de extração de medidas seguida de uma classificação e, as tarefas do segundo exemplo, como um problema de reconhecimento de padrões. A análise de imagens estuda meios para a resolução destas tarefas neste contexto abstrato. Entre algumas operações clássicas em análise de imagens podemos citar a segmentação e o reconhecimento de padrões.

- Compressão de Imagens : Quando uma imagem digital é criada, uma enorme quantidade de dados são gerados, muitas vezes comprometendo o armazenamento, transporte e processamento dos mesmos. Em compressão de imagens estudam-se meios para reduzir a quantidade de dados necessária para representar uma imagem, sem perda de informações.

- Síntese de Imagens : são operações que criam imagens a partir de outras imagens ou a partir de dados que não são imagens. Por exemplo, as imagens obtidas por tomografia são reconstituídas a partir de múltiplas projeções.

Cada um destes grupos contém uma diversidade de problemas e uma diversidade ainda maior de técnicas para a resolução dos mesmos. Entre as várias técnicas para resolver estes problemas, uma abordagem conhecida por Morfologia Matemática vem merecendo destaque recentemente (veja [Ser82], [HSZ87]).

1.2 Morfologia Matemática

Os estudos que originaram a teoria de Morfologia Matemática (*MM*) tiveram início em meados da década de sessenta, na *École Nationale Superiéure des Mines* de Paris, em *Fontainebleau*. Lá, George Matheron e Jean Serra lideravam um grupo que estudava a extração de informações de imagens a partir de transformações de formas. O estudo original realizado por Matheron consistia em analisar as estruturas geométricas das imagens microscópicas de amostras de metais e rochas, e relacioná-los com as propriedades físicas destes materiais.

As referidas transformações eram realizadas através de dois operadores elementares denominados dilatação e erosão, criados, respectivamente, a partir da noção de soma e subtração de Minkowski ([Min03] e [Had50, Had57]). Estes operadores eram parametrizados por elementos estruturantes, isto é, conjuntos cuja função consistia em fazer uma sondagem local numa vizinhança de cada ponto da imagem.

As aplicações destes operadores elementares permitiram verificar que a combinação destes resultava em operadores interessantes, os quais poderiam ainda ser combinados entre si para gerar operadores mais complexos. Verificou-se mais tarde que estes operadores eram estruturados sobre os reticulados completos.

Estas observações conduziram os pesquisadores aos resultados teóricos que foram inicialmente formalizados para o reticulado dos subconjuntos, abrangendo o domínio das imagens binárias. Posteriormente, os mesmos resultados foram estendidos para o reticulado das funções (que são utilizados para modelar o tratamento de imagens em níveis de cinza) e, em seguida, para o domínio dos reticulados completos quaisquer.

O ponto central da *MM* é a decomposição de quaisquer operadores definidos entre dois reticulados completos em termos de operadores elementares. Os primeiros resultados relacionados à decomposição de operadores foram apresentados por Matheron, em 1975 ([Mat75]). O Teorema de Representação de Matheron estabelece que qualquer operador entre dois reticulados Booleanos, invariante por translação (i.e, que não depende da localização) e crescente (i.e., que preserva a relação de ordem entre os reticulados), pode ser expresso como o supremo de um conjunto de erosões, ou dualmente, como o ínfimo de um conjunto de dilatações.

O Teorema da Decomposição Canônica, devido a Banon e Barrera, generaliza esse teorema para quaisquer operadores entre dois reticulados completos ([BB93]). Segundo este teorema, um operador pode ser expresso como o supremo de um conjunto de ínfimos entre uma erosão e outro operador elementar denominado anti-dilatação, ou dualmente, como o ínfimo de um conjunto de supremos entre uma dilatação e outro operador elementar denominado anti-erosão.

Os principais conceitos da *MM*, que formam a base desta teoria, encontram-se reunidos em três livros, [Mat75], [Ser82] e [Ser88], hoje considerados clássicos nesta área.

1.3 Máquinas Morfológicas

A característica de decomposição dos operadores citada acima permite que a *MM* possa ser interpretada como uma Linguagem Formal, na qual o vocabulário é formado pelos operadores elementares (erosão, dilatação, anti-erosão e anti-dilatação) e as operações de supremo e ínfimo ([BB92]). Em outras palavras, qualquer operador da *MM* pode ser representado através de uma frase desta linguagem.

Para definirmos uma linguagem formal, necessitamos definir uma gramática, ou seja, as regras de derivação das frases na linguagem, e uma semântica, isto é um modelo de interpretação das frases desta linguagem. Uma gramática e uma semântica para a linguagem formal considerada, que denominaremos Linguagem Morfológica (*LM*), são apresentadas, por exemplo, em [BB94], capítulo 8.

O estudo das linguagens formais compreende uma outra teoria, que deixaremos de abordar neste texto. Deve ficar claro, entretanto, que as frases da *LM* correspondem a todas as possíveis decomposições dos operadores da *MM* (uma destas possíveis decomposições é a decomposição canônica).

A *LM* é uma linguagem completa (isto é, qualquer operador da *MM* pode ser representado por uma frase da *LM*) e expressiva (isto é, com relativamente poucas palavras da *LM* podem-se expressar vários operadores interessantes da *MM*).

Portanto, o trabalho de resolver um problema de processamento de imagens pode ser encarado igualmente como o trabalho de encontrar uma frase da *LM* que aplicada sobre uma imagem realiza a sua transformação conforme o processamento desejado.

O mecanismo que possibilita a aplicação das frases morfológicas sobre as imagens são conhecidas como máquinas morfológicas (MMach's). Uma MMach é uma implementação particular da *LM*, isto é, é um hardware ou um software capaz de processar qualquer frase desta linguagem (em outras palavras, capaz de calcular eficientemente as erosões, dilatações, anti-erosões, anti-dilatações, supremos e ínfimos). A implementação de uma frase em uma MMach é denominada **programa**.

O esquema da figura 1.2 ilustra a relação entre os elementos citados acima.

1.4 Como Programar uma Máquina Morfológica ?

Em termos práticos, a resolução de problemas relacionados ao processamento de imagens, utilizando ferramentas da *MM*, corresponde a programar uma MMach. Mas, como se

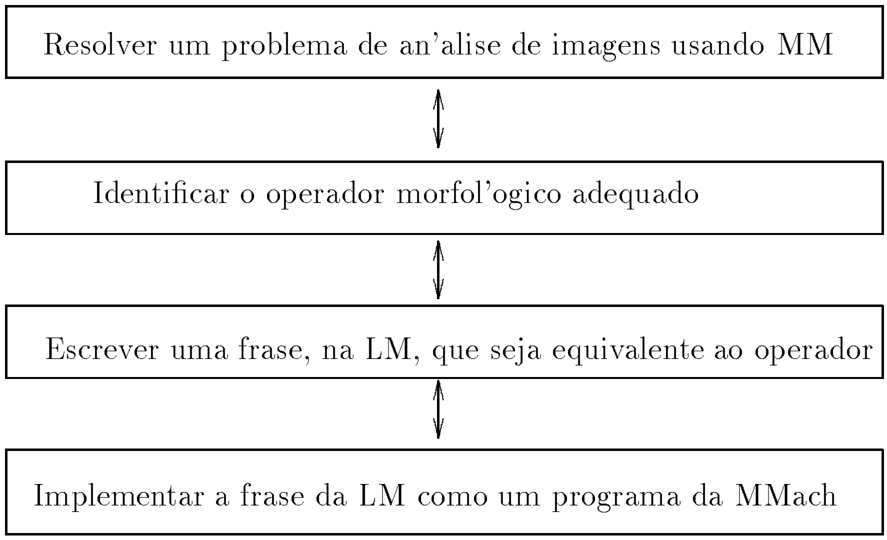


Figura 1.2: Formas equivalentes de encarar a resolução de problemas de processamento de imagens.

programa uma MMach ?

Em geral, o projeto destes programas é realizado de forma "ad hoc", onde a experiência e os conhecimentos específicos do usuário acerca de processamento de imagens e *MM* são de fundamental importância para que se consiga obter um bom resultado. Este fato restringe o universo de pessoas habilitadas para realizar este tipo de tarefa a um seleto grupo de especialistas. Esta não é uma situação muito interessante, uma vez que o processamento de imagens é uma atividade cada dia mais comum para diversos profissionais.

Face a esta situação, poderíamos recorrer a outras técnicas além da *MM* para resolver os problemas de processamento de imagens. No entanto, isto não elimina a necessidade da presença de um especialista. Além disso, devemos levar em consideração o fato de que a *MM*, diferentemente de várias outras técnicas, representa uma abordagem unificada para resolver vários problemas em análise de imagens, uma vez que qualquer operador entre duas imagens pode ser representado a partir da composição de operadores elementares ([Bar93]).

Estas observações sugerem a busca de métodos para automatizar a programação de MMach's. A característica de decomposição dos operadores morfológicos corrobora com esta sugestão, uma vez que permite pensarmos em projeto de operadores.

Um dos primeiros trabalhos conhecidos nesta linha de proposta é devido a E.R.Dougherty ([Dou92a, Dou92b]). Segundo a proposta de Dougherty, o operador morfológico é tratado como um estimador estatístico. A distribuição de probabilidade utilizada para estimar o filtro ótimo, segundo o critério do erro absoluto médio, é obtida a partir de pares de imagens que expressam a transformação desejada. Este trabalho será apresentado com maiores detalhes no capítulo 4.

Nos primeiros trabalhos de Dougherty, observa-se que o método por ele proposto apresenta um custo computacional elevado pois realiza buscas em um espaço de complexidade exponencial.

Este fato, mais a inexistência de outros métodos gerais para a programação de MMach's, levou-nos ao estudo deste assunto. Listamos a seguir, algumas características que julgamos importantes e desejáveis em um sistema para programação automática de MMach's :

- facilidade de uso - O sistema deve ser de fácil utilização em dois aspectos : 1) oferecer possibilidade de especificar o problema que se deseja resolver de forma bastante simples ; 2) não exigir conhecimentos específicos de Processamento de Imagens do usuário ;
- abrangência - O sistema deve representar uma solução abrangente, isto é, deve ser capaz de resolver vários tipos de problemas, não se restringindo às particularidades de alguns deles ;
- eficiência - O sistema deve ser computacionalmente eficiente ;
- precisão - O sistema deve produzir uma solução confiável, dentro de uma precisão pré-estabelecida ;
- fundamento teórico - O sistema deve estar baseado em resultados formais.

Restringimos os estudos a uma classe de operadores no domínio das imagens binárias, uma vez que, na prática, observa-se que vários problemas em análise de imagens podem ser resolvidos através da análise de imagens binárias. Em termos de *MM*, os resultados para o reticulado das partes de um conjunto são suficientes para modelar e tratar problemas associados às imagens binárias, conforme veremos no capítulo 3.

A proposta deste texto é exatamente apresentar uma metodologia para a programação automática de MMach's binárias. Em nossa proposta, apresentamos uma abordagem inédita baseado em técnicas de aprendizado através de exemplos. Em particular, estudamos o modelo de aprendizado "PAC" e suas extensões (capítulo 5).

1.5 Estrutura deste Texto

Este texto pode ser dividido em três partes principais. A primeira parte, que compreende os capítulos 2 ao 6, corresponde à exposição de resultados teóricos. A segunda parte, correspondente ao capítulo 7, representa uma das partes mais importantes do texto, na qual mostraremos como os resultados teóricos serão combinados a fim de especificar um modelo para programação automática de MMach's binárias. A terceira parte do texto consiste de três capítulos nos quais descrevemos a implementação do modelo proposto, vários exemplos de aplicação e análise dos resultados obtidos.

No capítulo 2, apresentaremos alguns conceitos relacionados à Teoria dos Conjuntos, Teoria dos Reticulados, Teoria das Probabilidades e noções de funções Booleanas. O intuito deste capítulo é reunir os elementos básicos utilizados no restante do texto, com o objetivo de proporcionar uma referência rápida, bem como para introduzir as notações e nomenclaturas utilizadas. Este capítulo foi introduzido para tornar o texto auto-contido.

No capítulo 3 apresentaremos as definições e resultados da Morfologia Matemática para subconjuntos, os quais são úteis para tratar problemas no domínio das imagens binárias. Um importante resultado deste capítulo são os Teoremas de Decomposição Canônica e Minimal. Mostraremos também que os operadores definidos entre subconjuntos são equivalentes às funções Booleanas e discutiremos algumas conseqüências interessantes decorrentes deste fato. Neste capítulo abordamos também a decomposição de operadores com propriedades especiais.

Quando as imagens tratadas são obtidas de um determinado contexto bem definido, elas possuem características que podem ser modeladas por processos aleatórios. No capítulo 4, descreveremos o conceito de "random sets" que possibilitam a modelagem das imagens no contexto estatístico, e descreveremos também o trabalho de Dougherty.

Independentemente à *MM*, um dos tópicos bastante estudados recentemente por pesquisadores da área de Inteligência Artificial, Redes Neurais e cientistas da computação em geral, são os processos de aprendizado (Machine Learning). O modelo de aprendizado *PAC* (do inglês "Probably Approximately Correct") proposto por Valiant em [Val84] e algumas extensões serão apresentados no capítulo 5. Estes serão utilizados para o aprendizado de conceitos (operadores) que podem ser modelados por funções Booleanas, conforme veremos no capítulo 7.

Os processos de aprendizado dependem de algoritmos denominados "algoritmos de aprendizado". No capítulo 6, introduziremos um algoritmo de aprendizado que pode também ser utilizado para minimização de funções Booleanas. Este algoritmo não foi encontrado na literatura consultada. Apresentaremos resultados comparativos entre o desempenho deste novo algoritmo e o algoritmo tabular de minimização de funções Booleanas de Quine-McCluskey.

Esses resultados teóricos são úteis para a especificação de um modelo para a programação automática de MMach's binárias, cuja descrição apresentamos no capítulo 7.

No capítulo 8 descrevemos a implementação, em ambiente UNIX, de um sistema baseado no modelo proposto. Vários experimentos foram realizados neste sistema, para ilustrar a abrangência do modelo. A descrição e a análise destes experimentos, que mostram várias aplicações possíveis, são apresentadas nos capítulos 9 e 10.

Finalmente, no capítulo 11, apresentaremos uma síntese deste trabalho e discutiremos algumas melhorias e extensões para o modelo, tendo em vista futuros rumos nesta área.

Capítulo 2

Noções Preliminares

Este capítulo constitui uma parte preliminar deste texto, no qual apresentaremos alguns elementos da Teoria dos Conjuntos, Teoria dos Reticulados, Teoria das Probabilidades e funções Booleanas.

As definições e conceitos aqui apresentados não pretendem esgotar toda a teoria, mas apenas abranger os elementos necessários nos capítulos posteriores. Também não serão apresentadas as demonstrações das propriedades e dos resultados citados, pois estas podem ser encontradas nas referências indicadas.

Acreditamos que o conteúdo deste capítulo pode ser facilmente compreendido e não representa obstáculo para o prosseguimento da leitura deste texto, pois consiste, em sua maioria, de definições e propriedades básicas. No entanto, é aconselhável uma rápida leitura do mesmo, para se obter familiarização com as notações e nomenclaturas utilizadas ao longo do texto.

2.1 Elementos da Teoria dos Conjuntos

Nesta seção apresentamos algumas definições e propriedades da teoria dos conjuntos ([Fil80]).

2.1.1 Conjuntos

Um **conjunto** A é uma coleção de elementos de um determinado universo, denotado genericamente por U. Se um elemento x de U **pertence** a A, escreve-se $x \in A$; caso contrário, escreve-se $x \notin A$.

Um conjunto pode ser representado enumerando-se todos os elementos que pertencem a ele, ou então, indicando-se as propriedades que devem satisfazer, como no seguinte exemplo.

Exemplo 2.1.1 *O conjunto A de todos os números inteiros ímpares maiores que zero e menores que 10 pode ser representado por* $A = \{1, 3, 5, 7, 9\}$ *ou* $A = \{x \in \mathbb{Z} : x$ *é ímpar e* $0 < x < 10\}$. $\diamond$

Um **conjunto vazio**, denotado por $\emptyset$, é um conjunto que não contém nenhum elemento. Um **conjunto finito** é um conjunto com um número finito de elementos. A **cardinalidade** de um conjunto A, denotada por $|A|$, corresponde ao número de elementos pertencentes a A.

Dois conjuntos A e B são **iguais** se e somente se todo elemento que pertence a um deles também pertence ao outro. Isto é, $A = B \Longleftrightarrow (x \in A \Longleftrightarrow x \in B, \quad \forall x)$. Eles são **diferentes** se e somente se existe ao menos um elemento que pertence a um deles e não pertence ao outro. Isto é, $A \neq B \Longleftrightarrow (\exists x \in A : x \notin B$ ou $\exists y \in B : y \notin A)$.

Um conjunto A está **contido** em B se e somente se todo elemento de A pertence a B, isto é, $A \subseteq B \Longleftrightarrow (x \in A \Longrightarrow x \in B, \ \forall x)$. Neste caso, diz-se que A é um **subconjunto** de B.

Um conjunto A está **propriamente contido** em B se e somente se A está contido em B e A é diferente de B, isto é, $A \subset B \Longleftrightarrow (A \subseteq B$ e $A \neq B)$. Neste caso, diz-se que A é um **subconjunto próprio** de B.

O **conjunto das partes de um conjunto** E, $\mathcal{P}(E)$, é o conjunto cujos elementos são todos os subconjuntos de E, isto é, $\mathcal{P}(E) = \{X : X \subseteq E\}$. Observe que $\emptyset \in \mathcal{P}(E)$, $E \in \mathcal{P}(E)$ e se $|E| = n$ então $|\mathcal{P}(E)| = 2^n$.

2.1.2 Operações com Conjuntos

Complemento

O **complemento** de um conjunto A, $A \subseteq B$, em relação a B é o conjunto de todos os elementos de B que não pertencem a A. Isto é, $A^c = \{x \in B : x \notin A\}$. Quando o conjunto B está claramente definido pelo contexto ou quando o complemento é em relação ao conjunto universo U, escreve-se simplesmente $A^c = \{x : x \notin A\}$.

Sejam $A, B \subseteq U$. Então, valem as seguintes propriedades (do complemento em relação a U) :

- $U^c = \emptyset$
- $\emptyset^c = U$

- $(A^c)^c = A$
- $A \subseteq B \Longrightarrow B^c \subseteq A^c$.

Interseção

A **interseção** de dois conjuntos A e B é o conjunto formado por todos os elementos que pertencem simultaneamente aos dois conjuntos. Isto é, $A \cap B = \{x : x \in A \text{ e } x \in B\}$.

A **interseção de** n **conjuntos** $A_1, A_2, ..., A_n$ é o conjunto dos elementos que pertencem simultaneamente a todos estes n conjuntos, isto é, $\bigcap_{i=1}^{n} A_i = \{x : x \in A_1 \text{ e } x \in A_2 \text{ e ... e } x \in A_n\}$.

Seja $\mathcal{X} \subseteq \mathcal{P}(E)$, uma coleção de subconjuntos de E. A **interseção de** $\mathcal{X}$ é o conjunto de todos os elementos x de E que satisfazem a propriedade : $x \in X$ para todo $X \in \mathcal{X}$, isto é, $\bigcap_{X \in \mathcal{X}} X = \{x \in E : x \in X, \ \forall X \in \mathcal{X}\}$.

Dois conjuntos A e B são **disjuntos** se e somente se $A \cap B = \emptyset$.

União

A **união** de dois conjuntos A e B é o conjunto formado por todos os elementos que pertencem a pelo menos um dos dois conjuntos. Isto é, $A \cup B = \{x : x \in A \text{ ou } x \in B\}$.

A **união de** n **conjuntos** $A_1, A_2, ..., A_n$ é o conjunto dos elementos que pertencem a pelo menos um destes n conjuntos, isto é, $\bigcup_{i=1}^{n} A_i = \{x : x \in A_1 \text{ ou } x \in A_2 \text{ ou ... ou } x \in A_n\}$.

Seja $\mathcal{X} \subseteq \mathcal{P}(E)$, uma coleção de subconjuntos de E. A **união de** $\mathcal{X}$ é o conjunto de todos os elementos x de E que possuem a propriedade : existe $X \in \mathcal{X}$ tal que $x \in X$, isto é, $\bigcup_{X \in \mathcal{X}} X = \{x \in E : \exists\ X, X \in \mathcal{X} \text{ e } x \in X\}$.

Propriedades da Interseção e da União

Seja U o conjunto universo e $A, B, C \subseteq U$. Valem as seguintes propriedades, cujas demonstrações podem ser encontradas por exemplo em [Fil80].

- $A \cap B \subseteq A$ e $A \cap B \subseteq B$
- $A \subseteq B \Longrightarrow A \cap B = A$
- $C \subseteq A$ e $C \subseteq B \Longrightarrow C \subseteq A \cap B$
- $A \subseteq B \Longrightarrow A \cap C \subseteq B \cap C$
- $\emptyset \cap A = \emptyset$
- $A \cap U = A$
- $A \cap A^c = \emptyset$
- $A \cap A = A$
- $A \cap B = B \cap A$
- $(A \cap B) \cap C = A \cap (B \cap C)$
- $x \in A \cap B \Longleftrightarrow x \in A$ e $x \in B$.

- $A \subseteq A \cup B$ e $B \subseteq A \cup B$
- $A \subseteq B \Longrightarrow A \cup B = B$
- $A \subseteq C$ e $B \subseteq C \Longrightarrow A \cup B \subseteq C$
- $A \subseteq B \Longrightarrow A \cup C \subseteq B \cup C$
- $\emptyset \cup A = A$
- $A \cup U = U$
- $A \cup A^c = U$
- $A \cup A = A$
- $A \cup B = B \cup A$
- $(A \cup B) \cup C = A \cup (B \cup C)$
- $x \in A \cup B \Longleftrightarrow x \in A$ ou $x \in B$.

2.1.3 Relação de Ordem em Conjuntos

Uma relação de ordem introduz uma estrutura no conjunto. As definições que se seguem serão importantes para trabalharmos com estruturas denominadas reticulados (que definiremos mais adiante).

Pares ordenados e n-uplas

Dados dois elementos x e y, um terceiro elemento que se indica por (x, y) chama-se **par ordenado**. O elemento x é a **primeira coordenada** e o elemento y é a **segunda coordenada** do par ordenado.

Genericamente, dados n elementos $x_1, x_2, ..., x_n$, um terceiro elemento indicado por $(x_1, x_2, ..., x_n)$ chama-se uma n**-upla ordenada**. Os elementos x_1, x_2, ..., x_n são respectivamente a **primeira, segunda**, ... e n**-ésima coordenada** da n-upla.

Produto cartesiano e potência de conjuntos

O **produto cartesiano** de A por B é o conjunto de todos os pares ordenados (x, y) tais que a primeira coordenada x pertence a A e a segunda coordenada y pertence a B, isto é $A \times B = \{(x, y) : x \in A \text{ e } y \in B\}$. O **quadrado de um conjunto** A é o produto cartesiano de A por A, isto é, $A^2 = A \times A = \{(x, y) : x, y \in A\}$.

Genericamente, o **produto cartesiano de** n **conjuntos** $A_1, A_2, ..., A_n$ é o conjunto de todas as n-uplas $(x_1, x_2, ..., x_n)$ tais que $x_1 \in A_1, x_2 \in A_2, ..., x_n \in A_n$, isto é, $A_1 \times A_2 \times ... \times A_n = \{(x_1, x_2, ..., x_n) : x_1 \in A_1, x_2 \in A_2, ..., x_n \in A_n\}$. No caso particular em que $A_1 = A_2 = ... = A_n = A$, $A^n = A_1 \times A_2 \times ... \times A_n$ é a n**-ésima potência de** A, isto é, $A^n = \{(x_1, x_2, ..., x_n) : x_1, x_2, ..., x_n \in A\}$.

Relações em um conjunto

Uma **relação** R de um conjunto A com um conjunto B é uma terna ordenada $R = (G, A, B)$, onde $G \subseteq A \times B$. Se $(x, y) \in G$, então diz-se que "x está relacionado a y pela relação R" e escreve-se xRy. Se $(x, y) \notin G$ então escreve-se $x \not R y$. Isto é,

$$(x, y) \in G \Longleftrightarrow xRy.$$

Quando $B = A$ dizemos simplesmente que R é uma relação em A.

Uma **relação de ordem** num conjunto não vazio A é uma relação $\leq = (G, A, A)$ que satisfaz as propriedades reflexiva, anti-simétrica e transitiva, isto é :

1) $x \in A \Longrightarrow x \leq x, \quad \forall x$ (reflexiva)

2) $x, y \in A$ e $x \leq y$ e $y \leq x \Longrightarrow x = y, \quad \forall x, y$ (anti-simétrica)

3) $x, y, z \in A$ e $x \leq y$ e $y \leq z \Longrightarrow x \leq z, \quad \forall x, y, z.$ (transitiva)

Dizemos que um conjunto A munido com uma relação de ordem $\leq$ é um **conjunto ordenado** pela relação de ordem $\leq$ e denotamos por $(A, \leq)$.

Seja $\leq$ uma relação de ordem em um conjunto não vazio E e seja A um subconjunto de E. A relação de ordem $\leq$ em E induz uma relação de ordem $\leq_A$ em A, definida por

$$(x, y \in A \text{ e } x \leq_A y) \Longleftrightarrow (x \leq y) \quad \forall x, y.$$

Dizemos que $\leq_A$ é a restrição da relação de ordem $\leq$ ao conjunto A, ou que a relação $\leq$ induz uma relação de ordem $\leq_A$ em A.

Conjuntos parcialmente ordenados

Dois elementos x e y são **comparáveis** se $x \leq y$ ou $y \leq x$. Caso contrário, eles não são comparáveis. Se $x \leq y$ e $x \neq y$, escrevemos também $x < y$. Pode-se também escrever $y \geq x$ quando $x \leq y$.

Um conjunto A com uma relação de ordem $\leq$ é **parcialmente ordenado**, ou um **poset** (do inglês "partialy ordered set"), se existe ao menos um par de elementos que não são comparáveis ([Bir67],[Fil80]). Se todos os elementos são comparáveis dois a dois, então o conjunto A é denominado um conjunto **totalmente ordenado**.

Seja $(A, \leq)$ um poset e sejam $a, b \in A$. Dizemos que "b cobre a" em A, se $a < b$ e não existe $x \in A$ tal que $a < x < b$. Posets finitos podem ser representados através de diagramas construídos da seguinte forma : os elementos de A são representados por pequenos círculos, dispostos de forma que o círculo para b está acima do círculo para a se $a < b$; no caso de b cobrir a, desenha-se um segmento de reta ligando a e b. A figura 2.1 mostra diagramas de alguns posets.

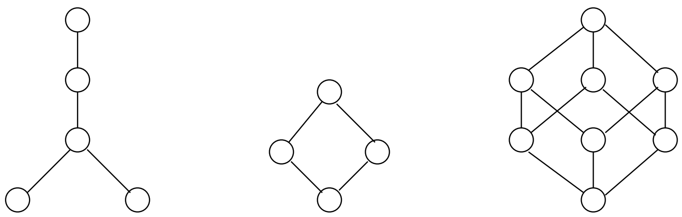


Figura 2.1: Diagrama de conjuntos parcialmente ordenados.

Exemplo 2.1.2 *A relação de inclusão $\subseteq$ no conjunto $\mathcal{P}(E)$, onde E é um conjunto com mais de um elemento, é uma ordem parcial, uma vez que se P e Q são dois elementos disjuntos de $\mathcal{P}(E)$, então eles não são comparáveis (isto é, $P \nsubseteq Q$ e $Q \nsubseteq P$).* $\diamond$

Intervalo fechado

Seja A um conjunto ordenado pela relação de ordem $\leq$ e sejam $a, b \in A$, $a \leq b$. O conjunto $[a, b] = \{x \in A : a \leq x \leq b\}$ é denominado **intervalo fechado** com **extremo esquerdo** a e **extremo direito** b.

2.1.4 Funções

Uma função ([Fil80]) é uma **relação binária** $f = (F, A, B)$, $F \subseteq A \times B$, se :

1) $\exists\ y, y \in B : x \in A \Longrightarrow (x, y) \in F,\ \forall x$;

2) $(x, y_1) \in F$ e $(x, y_2) \in F \Longrightarrow y_1 = y_2,\ \forall x$.

Uma função f de A em B é indicada por $f : A \rightarrow B$. Para cada elemento $x \in A$, o único elemento y de B tal que $(x, y) \in F$ é denominado **imagem de** x **pela função** f ou o **valor da função** f **no elemento** x e é denotado por $f(x)$. Um elemento genérico x de A é denominado uma **variável** e é geralmente denotado por uma letra minúscula do alfabeto (x, y, t, z, ...).

Os conjuntos A e B são denominados, respectivamente, **domínio** e **contradomínio** da função f. A **imagem** da função f é o conjunto formado por todos os elementos que são imagens de elementos de A. O conjunto de todas as funções de A em B é indicado por B^A.

Função sobrejetora

Uma função $f : A \to B$ é **sobrejetora** se e somente se para todo $y \in B$ existe $x \in A$ tal que $f(x) = y$, isto é,

$$y \in B \Longrightarrow (\exists\ x : x \in A \text{ e } f(x) = y), \qquad \forall y.$$

Função injetora

Uma função $f : A \to B$ é **injetora** se e somente se dois elementos distintos quaisquer x_1 e x_2 de A têm sempre imagens também distintas em B, isto é,

$$x_1, x_2 \in A \text{ e } x_1 \neq x_2 \Longrightarrow f(x_1) \neq f(x_2), \qquad \forall x_1, x_2.$$

Função bijetora

Uma função $f : A \to B$ é **bijetora** se e somente se ela é ao mesmo tempo injetora e sobrejetora.

Função Inversa

Dados A, B e $F \subseteq A \times B$, definimos $F^{-1} = \{(y, x) : (x, y) \in F\}$. A relação $f^{-1} = (F^{-1}, B, A)$ é uma função se e somente se $f = (F, A, B)$ é bijetora. A função f^{-1} é chamada a **função inversa** de f.

Toda função bijetora $f : A \to B$ e sua inversa $f^{-1} : B \to A$ estabelecem uma correspondência biunívoca entre A e B.

Composição de funções

Sejam A, B, C três conjuntos distintos ou não, $F_1 \subseteq A \times B$ e $F_2 \subseteq B \times C$. Definimos $F_2 \circ F_1 = \{(x, y) : \exists\ z, (x, z) \in F_1 \text{ e } (z, y) \in F_2\}$.

Sejam $f_1 : A \to B$ e $f_2 : B \to C$ duas funções. A relação $f_2 \circ f_1 = (F_2 \circ F_1, A, C)$ definida por

$$f_2 \circ f_1(x) = f_2(f_1(x)), \ \forall x \in A$$

é uma função de A em C denominada **função composta de** f_2 **e** f_1. Em geral, denotamos a composição de f_1 com f_2 por $f_2 f_1$ ou $f_2(f_1)$.

2.2 Elementos da Teoria dos Reticulados

Uma vez que um conjunto esteja dotado de uma relação de ordem, torna-se possível explorar esta relação para extrair vários resultados interessantes. A Teoria dos Reticulados ([Bir67, Szá63, Grä78]) estuda exatamente as propriedades e resultados decorrentes de conjuntos dotados com uma relação de ordem bem caracterizada.

2.2.1 Elementos Notáveis de um Conjunto Ordenado

Menor e maior elementos

Seja A um conjunto parcialmente ordenado pela relação de ordem $\leq$ e seja $X \subseteq A$.

$x \in A$ é o menor elemento de A ($minA$ ou O) se $x \leq y$ para todo $y \in A$, isto é,

$$O = minA = x \Longleftrightarrow (x \in A \text{ e } (y \in A \Longrightarrow x \leq y, \ \forall y)).$$

$x \in A$ é o maior elemento de A ($maxA$ ou I) se $y \leq x$ para todo $y \in A$, isto é,

$$I = maxA = x \Longleftrightarrow (x \in A \text{ e } (y \in A \Longrightarrow y \leq x, \ \forall y)).$$

Proposição 2.2.1 *O menor e maior elemento de A, se existirem, são únicos.* $\diamond$

Exemplo 2.2.1 *Para todo elemento $A \in \mathcal{P}(E)$, verifica-se $\emptyset \subseteq A \subseteq E$ e, portanto, o menor e o maior elemento de $\mathcal{P}(E)$ são respectivamente $\emptyset$ e E.* $\diamond$

Limitantes inferiores e superiores

Seja A um conjunto parcialmente ordenado pela relação de ordem $\leq$.

Todo elemento $a \in A$ tal que $a \leq x$ para todo $x \in X$ é denominado **limitante inferior** de X em A, isto é,

$$l.i.X = \{a \in A : a \leq x, \forall x \in X\}.$$

Todo elemento $a \in A$ tal que $x \leq a$ para todo $x \in X$ é denominado **limitante superior** de X em A, isto é,

$$l.s.X = \{a \in A : x \leq a, \forall x \in X\}.$$

Ínfimo e supremo

Seja A um conjunto parcialmente ordenado pela relação de ordem $\leq$ e seja $X \subseteq A$.

O **ínfimo** de X em A ($\wedge X$) é o maior elemento do conjunto dos limitantes inferiores de X em A, isto é,

$$\wedge X = max(l.i.X).$$

O **supremo** de X em A ($\vee X$) é o menor elemento do conjunto dos limitantes superiores de X em A, isto é,

$$\vee X = min(l.s.X).$$

Exemplo 2.2.2 *No conjunto $\mathcal{P}(E)$ das partes de E, um subconjunto $\mathcal{X} \subseteq \mathcal{P}(E)$ possui ínfimo e supremo que são respectivamente $\bigcap_{X \in \mathcal{X}} X$ e $\bigcup_{X \in \mathcal{X}} X$.* $\diamond$

2.2.2 Reticulados

Definição 2.2.1 *Um conjunto parcialmente ordenado L é um* **reticulado** *se e somente se qualquer conjunto de dois elementos x e y possui ínfimo e supremo em L, denotados respectivamente por $x \wedge y$ e $x \vee y$.* $\diamond$

Definição 2.2.2 *Um* **subreticulado** *de um reticulado L é um conjunto $A \subseteq L$ tal que $a \wedge b \in A$ e $a \vee b \in A$, $\forall a, b \in A$.* $\diamond$

Um reticulado L é :

- **completo** se e somente se qualquer subconjunto de L possui um ínfimo e um supremo em L.

  Qualquer reticulado finito é um reticulado completo.

- **distributivo** se e somente se $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z), \quad \forall x, y, z \in L$.

- **complementado** se e somente se todos os seus elementos possuem complemento.

  O **complemento** de um elemento x em um reticulado L com O e I é um elemento $y \in L$ tal que $x \wedge y = O$ e $x \vee y = I$, e é denotado por x^c.

- **Booleano** se e somente se ele é distributivo e complementado.

Exemplo 2.2.3 *O conjunto $(\mathcal{P}(E), \subseteq)$ é um reticulado completo onde $O = \emptyset$ e $I = E$. Para qualquer subconjunto $\mathcal{X} \subseteq \mathcal{P}(E)$, $inf\mathcal{X} = \bigcap_{X \in \mathcal{X}} X$ e $sup\mathcal{X} = \bigcup_{X \in \mathcal{X}} X$, e para quaisquer $X, Y \in \mathcal{P}(E)$, $X \wedge Y = X \cap Y$ e $X \vee Y = X \cup Y$.* $\diamond$

Exemplo 2.2.4 *O conjunto* $(\mathcal{P}(E), \subseteq)$ *é um reticulado Booleano pois ele é distributivo (isto é,* $X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$, $\forall X, Y, Z \in \mathcal{P}(E)$*) e é complementado (isto é,* $X \cap X^c = \emptyset$ *e* $X \cup X^c = E$, $\forall X \in \mathcal{P}(E)$*).* ◇

Princípio da Dualidade

Seja $(L, \leq)$ um poset. Então o conjunto $(L, \geq)$ onde $a \geq b$ significa $b \leq a$ é também um poset, e é denominado dual do poset $(L, \leq)$.

Uma proposição de reticulado é uma expressão lógica P composta por termos lógicos (por exemplo, "igual", "para cada elemento do reticulado ... vale", "se e somente se", etc), variáveis (a, b, ...) e os símbolos $\vee$ e $\wedge$.

Exemplo 2.2.5 *As seguintes expressões são proposições de reticulado :*

- *Para quaisquer elementos a, b, c de um reticulado,* $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
- *Para quaisquer elementos a e b de um reticulado,* $a \wedge b = a \vee b$ *se e somente se* $a = b$.

◇

O **dual** de uma proposição de reticulado P, denotado por $\mathcal{D}(P)$, é obtido trocando-se $\vee$ por $\wedge$ e vice-versa. Observe que $\mathcal{D}(\mathcal{D}(P)) = P$.

Exemplo 2.2.6 *As seguintes expressões são proposições de reticulado, duais às proposições do exemplo anterior :*

- *Para quaisquer elementos a, b, c de um reticulado,* $(a \vee b) \vee c = a \vee (b \vee c)$
- *Para quaisquer elementos a e b de um reticulado,* $a \vee b = a \wedge b$ *se e somente se* $a = b$

◇

Princípio da Dualidade : O dual $\mathcal{D}(P)$ de qualquer proposição de reticulado P verdadeira é também uma proposição de reticulado verdadeira.

O princípio da dualidade pode ser utilizado para demonstrar teoremas (veja por exemplo, [Szá63], página 36).

Teorema 2.2.1 (De Morgan) *Em um reticulado Booleano vale* $(a \wedge b)^c = a^c \vee b^c$ *e* $(a \vee b)^c = a^c \wedge b^c$. ◇

2.2.3 Isomorfismo de Reticulados

Sejam A e B dois reticulados ordenados pelas relações de ordem $\leq_A$ e $\leq_B$ respectivamente. Seja $f : A \rightarrow B$ uma função bijetora de A em B. A função f é um **isomorfismo** de A em B se e somente se, quaisquer que sejam x e y em A, tem-se :

$$x \leq_A y \Longleftrightarrow f(x) \leq_B f(y).$$

2.3 Elementos da Teoria de Probabilidades

Os modelos matemáticos apropriados para o estudo de fenômenos observáveis são, em muitos casos, **não-determinísticos** ou **probabilísticos**. Além disso, os métodos estatísticos utilizados pela Inferência Estatística são baseados em considerações probabilísticas. A fim de empregar adequadamente as técnicas estatísticas torna-se também necessário, portanto, o estudo da Teoria de Probabilidades ([Mey69, Bre87]).

2.3.1 Espaço de Probabilidades

O principal elemento da Teoria de Probabilidades é o **espaço de probabilidade**. Um espaço de probabilidade associado a um experimento ([Mey69]) é uma terna $(\Omega, \mathcal{F}, P)$, onde

Ω é o espaço amostral, isto é, o conjunto de todos os possíveis resultados de um experimento,

$\mathcal{F}$ é o conjunto de eventos, isto é, uma coleção de subconjuntos de Ω,

P é uma função de $\mathcal{F}$ em $[0, 1]$ que associa a cada evento $A \in \mathcal{F}$ uma probabilidade $P(A)$.

O conjunto de eventos $\mathcal{F}$ é uma σ-álgebra, isto é :

- $\Omega \in \mathcal{F}$
- se $A \in \mathcal{F}$, então $A^c \in \mathcal{F}$.
- se uma seqüência de eventos $A_1, A_2, ..., A_n$, $n \geq 1$, tem todos os seus membros em $\mathcal{F}$, então $\cup_{i=1}^{n} A_i \in \mathcal{F}$.

A função $P : \mathcal{F} \rightarrow [0, 1]$ é uma função de probabilidade se satisfaz os seguintes axiomas :

- $\forall A \in \mathcal{F}, 0 \leq P(A) \leq 1$,
- $P(\Omega) = 1$,
- se $A \cap B = \emptyset \Longrightarrow P(A \cap B) = P(A)P(B)$,
- se $A_1, A_2, ..., A_n$ são dois a dois mutuamente exclusivos, então $P(\cup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$.

Observações : $\emptyset$ e Ω são chamados respectivamente evento impossível e evento certo. Dois eventos A e B são mutuamente exclusivos se $A \cap B = \emptyset$. Se Ω é finito ou infinito enumerável, todo subconjunto de Ω poderá ser considerado um evento.

Exemplo 2.3.1 *Considere um experimento que consiste no lançamento de uma moeda e na observação da face voltada para cima. Sejam C a face cara e O a face coroa da moeda. O espaço amostral neste caso é $\Omega = \{C, O\}$. Os possíveis eventos são $\emptyset$, $\{C\}$, $\{O\}$ e Ω. Supondo que a moeda não está viciada (isto é, não tem maior tendência de cair com uma determinada face para cima) é razoável supor que $P(C) = P(O) = \frac{1}{2}$ e $P(\emptyset) = 0$. O evento Ω corresponde ao evento "a face observada é C ou O", e portanto $P(\Omega) = 1$.* $\diamond$

Probabilidade Condicionada

A probabilidade de um evento A, condicionada ao evento B, é definida por

$$P(A/B) = \frac{P(A \cap B)}{P(B)}$$

e lê-se "probabilidade de A dado B".

2.3.2 Variáveis Aleatórias

Uma função $\boldsymbol{x}$ que associa a cada elemento $s \in \Omega$ um número real, $\boldsymbol{x}(s)$, é denominada uma **variável aleatória**.

O contradomínio $R_{\boldsymbol{x}}$ de $\boldsymbol{x}$ são todos os possíveis valores que a função $\boldsymbol{x}$ pode tomar. Se $R_{\boldsymbol{x}}$ é finito ou infinito enumerável, dizemos que $\boldsymbol{x}$ é uma variável aleatória discreta.

Exemplo 2.3.2 *Considere o experimento que consiste no lançamento consecutivo de duas moedas e na observação da face voltada para cima. Relativamente a este experimento, o espaço amostral será $\Omega = \{CC, CO, OC, OO\}$. Suponha que queremos avaliar o número* *de caras nos lançamentos. Seja $\boldsymbol{x}(s)$ = "número de caras na amostra s". $\boldsymbol{x}$ é uma variável aleatória discreta e seu contradomínio é $R_{\boldsymbol{x}} = \{0, 1, 2\}$.* ◇

Seja P uma função de probabilidade associada a Ω. Dado um evento A, $P(A)$ é a probabilidade associada a A. A função de probabilidade P induz uma função de probabilidade $P_{\boldsymbol{x}}$ em $R_{\boldsymbol{x}}$, definida por :

$$P_{\boldsymbol{x}}(B) = P(A), \text{ onde } A = \{s \in \Omega : \boldsymbol{x}(s) \in B\}.$$

Distribuição de Probabilidade

Seja $\boldsymbol{x}$ uma variável aleatória discreta com contradomínio finito, isto é, $R_{\boldsymbol{x}} = \{x_1, x_2, ..., x_n\}$. Se para cada ponto x_i de $R_{\boldsymbol{x}}$ definirmos sua probabilidade como $p(x_i) = P[\boldsymbol{x} = x_i]$, teremos um espaço de probabilidade. A função p é denominada **distribuição ou função de probabilidade**.

A função p satisfaz as condições

1. $p(x_i) \geq 0, \forall i$,
2. $\sum_{i=1}^{n} p(x_i) = 1$.

Função de Distribuição Acumulada

Seja $\boldsymbol{x}$ uma variável aleatória discreta. A função $F(x) = P(\boldsymbol{x} \leq x)$ é denominada **função de distribuição acumulada** da variável aleatória $\boldsymbol{x}$. É fácil verificar que

$$F(x) = \sum_j \{p(x_j) : x_j \leq x\}.$$

Esperança de uma variável aleatória

Seja $\boldsymbol{x}$ uma variável aleatória discreta com contradomínio finito e seja p a sua distribuição de probabilidade. Então, a **esperança** de $\boldsymbol{x}$, denotada por $E(\boldsymbol{x})$ é definida por

$$E[\boldsymbol{x}] = \sum_i x_i p(x_i).$$

A **esperança condicional** de $\boldsymbol{x}$, dado $\boldsymbol{y} = y$, é definida por

$$E[\boldsymbol{x}/\boldsymbol{y} = y] = \sum_i x_i p(x_i/y).$$

Variáveis Aleatórias n-dimensionais

Uma n-upla de variáveis aleatórias discretas $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ é denominada um vetor aleatório discreto n-dimensional. Uma função $p : (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n) \to [0, 1]$ satisfazendo

1. $p(x_1, x_2, ..., x_n) \geq 0$, para toda realização $(x_1, x_2, ..., x_n)$ de $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$
2. $\sum_{i_n} ... \sum_{i_1} p(x_{i_1}, ..., x_{i_n}) = 1$.

é a função de probabilidade de $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$.

Sejam duas variáveis aleatórias discretas $\boldsymbol{x}$ e $\boldsymbol{y}$. Então $\boldsymbol{x}$ e $\boldsymbol{y}$ são independentes se

$$P(\boldsymbol{x} = x_i, \boldsymbol{y} = y_i) = P(\boldsymbol{x} = x_i)P(\boldsymbol{y} = y_i), \text{para toda realização } (x_i, y_i) \text{ de } (\boldsymbol{x}, \boldsymbol{y}).$$

A função p definida por $p(x_i, y_i) = P(\boldsymbol{x} = x_i, \boldsymbol{y} = y_i)$, para cada realização (x_i, y_i) de $(\boldsymbol{x}, \boldsymbol{y})$, é a distribuição de probabilidade conjunta de $(\boldsymbol{x}, \boldsymbol{y})$.

2.4 Noções de Funções Booleanas

Os sistemas digitais, particularmente os computadores, executam processos de decisões lógicas implementados através de circuitos lógicos. Estes processos são modelados por funções Booleanas e por esta razão, o estudo das funções Booleanas apresenta-se fortemente associado à Teoria dos Circuitos de Chaveamento ([HP81]).

Do ponto de vista puramente matemático, a técnica utilizada para o projeto de circuitos lógicos faz parte de uma teoria conhecida por Álgebra Booleana. Nesta seção, apresentaremos as funções Booleanas dentro deste contexto estritamente matemático ([Rut65]).

2.4.1 Álgebra Booleana

Uma álgebra Booleana $\mathcal{B} = (B, +, \cdot, \bar{\cdot})$ é um conjunto de elementos B com duas operações binárias, $+$ e $\cdot$ (denominados, respectivamente, soma e produto), para os quais valem os seguintes axiomas.

- Axioma 1 : As operações $+$ e $\cdot$ são comutativas.

  $a + b = b + a, \ \forall a, b \in B$ e

  $a \cdot b = b \cdot a, \ \forall a, b \in B$

- Axioma 2 : Existem dois elementos identidade, 0 e 1, com respeito, respectivamente, às operações $+$ e $\cdot$.

  $a + 0 = a, \forall a \in B$

  $a \cdot 1 = a, \forall a \in B$

- Axioma 3 : Cada operação é distributiva relativamente a outra.

  $a + (b \cdot c) = (a + b) \cdot (a + c), \ \forall a, b \in B$

  $a \cdot (b + c) = (a \cdot b) + (a \cdot c), \ \forall a, b \in B$

- Axioma 4 : Para cada elemento $a \in B$, existe um elemento inverso $\overline{a} \in B$ tal que $a \cdot \overline{a} = 0$ e $a + \overline{a} = 1$.

Na tabela 2.1 listamos os principais teoremas e leis da Álgebra Booleana.

Lei ou Teorema	Operação $+$	Operação $\cdot$
Operação com 0 ou 1	$x + 0 = x$ $x + 1 = 1$	$x \cdot 1 = x$ $x \cdot 0 = 0$
Teorema da idempotência	$x + x = x$	$x \cdot x = x$
Teorema da Involução	$\overline{\overline{x}} = x$	
Teorema da Complementação	$x + \overline{x} = 1$	$x \cdot \overline{x} = 0$
Lei da Comutatividade	$x + y = y + x$	$x \cdot y = y \cdot x$
Lei da Associatividade	$(x + y) + z = x + (y + z)$ $= x + y + z$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$ $= x \cdot y \cdot z$
Lei Distributiva	$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
Teoremas de simplificação	$x \cdot y + x \cdot \overline{y} = x$ $x + x \cdot y = x$ $(x + \overline{y}) \cdot y = x \cdot y$	$(x + y) \cdot (x + \overline{y}) = x$ $x \cdot (x + y) = x$ $(x \cdot \overline{y}) + y = x + y$
Leis de DeMorgan	$(\overline{x + y + z + ...}) =$ $\overline{x} \cdot \overline{y} \cdot \overline{z} \cdot ...$	$(\overline{x \cdot y \cdot z \cdot ...}) =$ $\overline{x} + \overline{y} + \overline{z} + ...$

Tabela 2.1: Principais Leis e Teoremas da Álgebra Booleana

Expressões Booleanas

Sejam $x_1, x_2, \ldots, x_n$ variáveis que tomam valor em B. Uma expressão Booleana é qualquer expressão consistindo de uma variável ou construída a partir destas variáveis aplicando-se, um número finito de vezes, as operações $+$, $\cdot$ e $\bar{\cdot}$.

As seguintes expressões são expressões Booleanas com três variáveis, x_1, x_2 e x_3. Observe que o símbolo $\cdot$ correspondente ao operador produto foi omitido (portanto, em vez de $\overline{x}_1 \cdot \overline{x}_2$ escrevemos simplesmente $\overline{x}_1\overline{x}_2$).

$$\overline{x}_1\overline{x}_2 + x_1(\overline{x}_2 + x_2\overline{x}_3) \tag{2.1}$$

$$\overline{x}_1\overline{x}_2 + x_1\overline{x}_2 + x_1x_2\overline{x}_3 \tag{2.2}$$

$$x_2 + x_1\overline{x}_3 \tag{2.3}$$

Equivalência e Simplificação de Expressões Booleanas

Definição 2.4.1 *Duas expressõess Booleanas são* **equivalentes** *se e somente se uma delas pode ser deduzida a partir da outra através da aplicação das leis e teoremas da álgebra Booleana, um número finito de vezes.* ◇

Denominamos **literal** qualquer uma das variáveis ou a respectiva variável barrada (x_1, $\overline{x}_1$, x_2, $\overline{x}_2$, ...). Um **produto** é uma série de literais relacionados pelo operador produto ($x_1 \cdot x_2$, $x_2 \cdot \overline{x}_3 \cdot x_4$, ...). Uma **soma** é uma série de literais relacionados pelo operador soma ($x_1 + x_2 + x_3$, $x_2 + x_5$, ...).

A simplificação de uma expressão Booleana consiste de qualquer processo que produz uma outra expressão equivalente contendo menor número de termos ou literais. A expressão resultante de um processo de simplificação é denominada uma expressão simplificada. As expressões Booleanas podem ser simplificadas segundo as Leis e Teoremas da Álgebra Booleana (ver por exemplo, [HP81], seção 4.4).

Exemplo 2.4.1 *Neste exemplo mostramos a simplificação da expressão* $\overline{x}_1\overline{x}_2 + x_1\overline{x}_2 + x_1x_2\overline{x}_3$ *:*

$$\begin{array}{rr}
\overline{x}_1\overline{x}_2 + x_1\overline{x}_2 + x_1x_2\overline{x}_3 & \textit{(Lei distributiva)} \\
\overline{x}_2(\overline{x}_1 + x_1) + x_1x_2\overline{x}_3 & \textit{(Lei distributiva)} \\
\overline{x}_2(1) + x_1x_2\overline{x}_3 & \textit{(Complementação)} \\
\overline{x}_2 + x_1x_2\overline{x}_3 & \\
\overline{x}_2 + x_1\overline{x}_3 & \textit{(Teoremas de simplificação)}
\end{array}$$

Uma vez que a expressão 2.3 pode ser deduzida de 2.2 através de simplificações, eles são equivalentes. ◇

Forma Canônica de Expressões Booleanas

Um produto em que aparecem exatamente n literais, no qual o literal x aparece se, e somente se, o seu complemento $\overline{x}$ não aparece, e vice-versa, é denominado **produto**

canônico. Analogamente, uma soma na qual o literal x aparece se, e somente se, o seu complemento $\overline{x}$ não aparece, e vice-versa, é denominado uma **soma canônica**.

FND e FNC : Qualquer função Booleana pode ser expressa em termos de soma de produtos canônicos (**forma normal disjuntiva ou FND**) ou, em termos de produto de somas canônicas (**forma normal conjuntiva ou FNC**) (veja, por exemplo, [HP81], capítulo 6).

As seguintes expressões correspondem, respectivamente, às formas FND e FNC da expressão 2.3.

$$\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2 x_3 + x_1 x_2\overline{x}_3 \tag{2.4}$$
$$(x_1 + \overline{x}_2 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) \tag{2.5}$$

2.4.2 Um Exemplo de Álgebra Booleana

Pode-se facilmente verificar que o conjunto $B = \{0, 1\}$ com as operações $+$, $\cdot$ e $\bar{\cdot}$ definidas na tabela 2.2 é uma álgebra Booleana. Observe que as operações $+$, $\cdot$ e $\bar{\cdot}$ correspondem respectivamente aos operadores OR, AND e NOT da teoria dos circuitos de chaveamento ([Rut65]).

a	b	$a+b$	$a \cdot b$	$\overline{a}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Tabela 2.2: Definição das operações $+$, $\cdot$ e $\bar{\cdot}$ em $B = \{0, 1\}$.

2.4.3 Funções Booleanas

A partir deste ponto, consideramos a álgebra Booleana $B = (\{0, 1\}, +, \cdot, \bar{\cdot})$.

Definição 2.4.2 (Preliminar) *Uma* **função Booleana** *é uma função que mapeia um certo número n de variáveis que tomam valor em $\{0, 1\}$ sobre o conjunto $\{0, 1\}$. Estas n variáveis, denotadas por $x_1, x_2, \ldots, x_n$, são denominadas* **variáveis Booleanas**. $\diamond$

Uma outra forma de definirmos funções Booleanas é associando cada variável Booleana a um ponto de um conjunto, da seguinte forma. Seja W um conjunto finito, de cardinalidade n. A cada um dos elementos de W associamos uma variável x que toma valor 1, relativamente a um subconjunto $X \subseteq W$, se e somente se o correspondente elemento de W pertence a X; caso contrário, x toma o valor 0.

Usando a mesma notação, x, tanto para indicar a variável como o seu correspondente elemento em W, temos $x = 1 \iff x \in X$.

Definição 2.4.3 *Uma função Booleana é uma função definida de $\mathcal{P}(W)$ em $\{0,1\}$. Se $|W| = n$, então a função Booleana possui n variáveis denotadas por $x_1, x_2, \ldots, x_n$.* ◇

O conjunto de todas as funções de $\mathcal{P}(W)$ em $\{0,1\}$ é indicado por $\{0,1\}^{\mathcal{P}(W)}$ e $|\{0,1\}^{\mathcal{P}(W)}| = 2^{2^n}$.

Avaliação de expressões Booleanas

Exemplo 2.4.2 *Levando-se em conta que $0 + x = 0$, $1 + x = 1$, $0x = 0$ e $1x = x$, o valor da expressão 2.1, para $x_1 = 1, x_2 = 0$ e $x_3 = 0$, pode ser facilmente avaliado :*

$$\overline{x}_1\overline{x}_2 + x_1(\overline{x}_2 + x_2\overline{x}_3) = 0 \cdot 1 + 1(1 + 0 \cdot 1) = 0 + 1(1 + 0) = 0 + 1(1) = 0 + 1 = 1.$$

◇

Tabelas-verdade

A tabela-verdade de uma função Booleana f consiste de uma tabela formada por duas colunas : a primeira coluna é preenchida pelas 2^n possíveis formas de associarmos valores às n variáveis e a segunda coluna pelo correspondente valor da função f.

Na primeira coluna da tabela 2.3a estão relacionadas todas as $2^3 = 8$ possíveis formas de associação de valores às 3 variáveis x_1, x_2 e x_3 e na segunda coluna está relacionado o valor que a função $\overline{x}_1\overline{x}_2 + x_1(\overline{x}_2 + x_2\overline{x}_3)$ toma em cada uma destas associações .

Uma função Booleana define e é totalmente definida pela sua *tabela-verdade*. Portanto, duas expressões Booleanas são equivalentes se e somente se as correspondentes tabelas-verdade são iguais. As expressões 2.1 e 2.3 são equivalentes pois as suas tabelas-verdade (2.3a e 2.3b, respectivamente) são iguais.

Simplificação de notação

O conjunto de todas as seqüências de n bits correspondem justamente à representação binária dos números entre 0 e $2^n - 1$. Com base neste fato, podem-se caracterizar os

$x_1x_2x_3$	f_1
000	1
001	1
010	0
011	0
100	1
101	1
110	1
111	0

(a)

$x_1x_2x_3$	f_3
000	1
001	1
010	0
011	0
100	1
101	1
110	1
111	0

(b)

Tabela 2.3: Tabela-verdade. (a) $f_1 = \overline{x}_1\overline{x}_2 + x_1(\overline{x}_2 + x_2\overline{x}_3)$. (b) $f_3 = \overline{x}_2 + x_1\overline{x}_3$.

produtos canônicos (que serão denominados **mintermos**), e as somas canônicas (que serão denominadas **maxtermos**) através da notação decimal correspondente. A tabela 2.4 apresenta os mintermos e maxtermos para três variáveis e a notação associada a cada um deles.

$x_1x_2x_3$	maxtermos	mintermos
0 0 0	$x_1 + x_2 + x_3 = M_0$	$\overline{x}_1\overline{x}_2\overline{x}_3 = m_0$
0 0 1	$x_1 + x_2 + \overline{x}_3 = M_1$	$\overline{x}_1\overline{x}_2x_3 = m_1$
0 1 0	$x_1 + \overline{x}_2 + x_3 = M_2$	$\overline{x}_1x_2\overline{x}_3 = m_2$
0 1 1	$x_1 + \overline{x}_2 + \overline{x}_3 = M_3$	$\overline{x}_1x_2x_3 = m_3$
1 0 0	$\overline{x}_1 + x_2 + x_3 = M_4$	$x_1\overline{x}_2\overline{x}_3 = m_4$
1 0 1	$\overline{x}_1 + x_2 + \overline{x}_3 = M_5$	$x_1\overline{x}_2x_3 = m_5$
1 1 0	$\overline{x}_1 + \overline{x}_2 + x_3 = M_6$	$x_1\overline{x}_2x_3 = m_6$
1 1 1	$\overline{x}_1 + \overline{x}_2 + \overline{x}_3 = M_7$	$x_1x_2x_3 = m_7$

Tabela 2.4: Tabela de maxtermos e mintermos

Observe que $x_1\overline{x}_2x_3 = 1 \iff x_1 = 1, x_2 = 0$ e $x_3 = 1$. Portanto, a FND de uma expressão booleana pode ser diretamente obtida através da soma dos mintermos correspondentes aos 1's da sua tabela-verdade. Analogamente, $(x_1 + \overline{x}_2 + x_3) = 0 \iff x_1 = 0, x_2 = 1$ e $x_3 = 0$, e portanto, a FNC pode ser obtida pelo produto dos maxtermos correspondentes aos 0's da tabela-verdade (compare a tabela 2.3a com as expressões 2.4 e 2.5).

A partir da tabela 2.3a e 2.4 deriva-se facilmente a FND da função $\overline{x}_1\overline{x}_2 + x_1(\overline{x}_2 + x_2\overline{x}_3)$ que é dada por

$$f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1\overline{x}_2x_3$$

e sua notação simplificada é dado por :

$$f(x_1, x_2, x_3) = \sum m(0, 1, 4, 5, 6).$$

Na FNC, $f(x_1, x_2, x_3) = (x_1 + \overline{x}_2 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) = \prod M(2, 3, 7)$.

2.4.4 Representação Cúbica de uma Função Booleana

As seqüências de n bits, correspondentes a todas as possíveis combinações de valores que as n variáveis de uma função Booleana podem tomar, podem ser representadas por pontos no n-espaço. A coleção de todos os 2^n pontos possíveis formam os vértices de um ***n*-cubo** ([HP81], capítulo 7). A figura 2.2 mostra um 3-cubo.

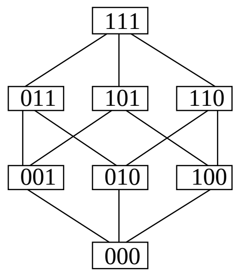


Figura 2.2: Diagrama do 3-cubo.

Os vértices de um n-cubo são denominados 0-cubos. Dois 0-cubos formam um 1-cubo se eles diferem em apenas uma coordenada. Quatro 0-cubos formam um 2-cubo se eles são iguais a menos de duas coordenadas. De modo geral, 2^k 0-cubos formam um k-cubo se eles são iguais a menos de k coordenadas.

Denotamos um k-cubo colocando um X nas coordenadas que não são iguais. Assim, o 1-cubo $\{000, 100\}$ é representado por $X00$. O 2-cubo $\{000, 001, 100, 101\}$ é representado por $X0X$.

Exemplo 2.4.3 *Na figura 2.3, dizemos que o* 0*-cubo* 000 *está* contido *no (ou é coberto*[1] *pelo)* 1*-cubo* $X00$*, ou ainda, que o* 1*-cubo* $X00$ cobre *o* 0*-cubo* 000*. Analogamente, dizemos que o* 1*-cubo* $X00$ *está contido no* 2*-cubo* $X0X$ *ou que o* 2*-cubo* $X0X$ *cobre o cubo* $X00$*.* $\diamond$

Dizemos que um k-cubo tem **dimensão** k. Dado um conjunto V de vértices de um n-cubo, dizemos que um k-cubo formado pelos vértices de V é **maximal** se não existe outro cubo de dimensão maior também formado por vértices de V, e que contenha o k-cubo.

[1] o termo "cobrir" utilizado aqui difere do utilizado na página 15 para expressar a relação entre dois elementos de um poset.

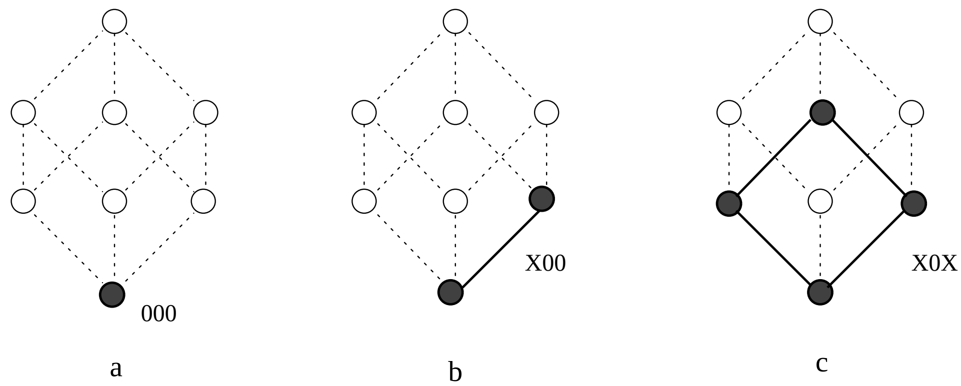


Figura 2.3: O 3-cubo e alguns sub-cubos.

Definição 2.4.4 *A* **representação cúbica** *de uma função Booleana f de n variáveis consiste do conjunto de vértices do n-cubo que correspondem aos mintermos de f.* ◇

Definição 2.4.5 *A* **representação cúbica minimal** *de uma função Booleana de n variáveis consiste do conjunto de todos os cubos maximais formados por vértices do n-cubo que correspondem aos seus mintermos.* ◇

Exemplo 2.4.4 *A função Booleana $f = \sum m(0,1,4,5,6)$ é representada pelos vértices* 000, 001, 100, 101 *e* 110. *A sua representação cúbica e minimal são apresentadas graficamente, respectivamente, nas figura 2.4a e 2.4b.* ◇

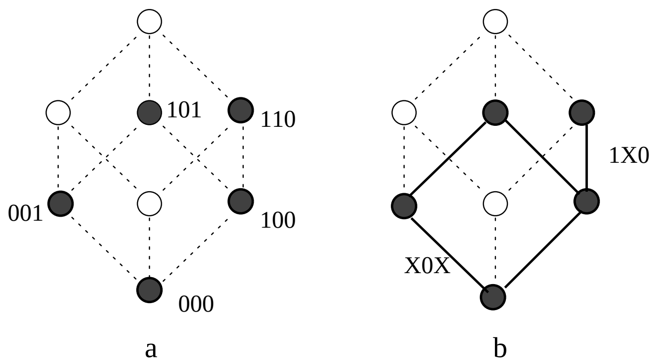


Figura 2.4: Representação (a) cúbica e (b) minimal da função $f = \sum m(0,1,4,5,6)$.

Intervalos

O n-cubo forma uma estrutura de reticulado Booleano. Portanto, podemos associar a um cubo a noção de *intervalo*, cujo extremo esquerdo e direito são respectivamente o menor e o maior elemento do cubo. Ao cubo $10X$ associamos o intervalo $[100, 101] = \{x \in \{0,1\}^3 : 100 \leq x \leq 101\}$ e ao cubo $1XX$ associamos o intervalo $[100, 111]$. A *dimensão de um intervalo* $[A, B]$, denotado $dim([A, B])$, é definida como a dimensão do cubo associado a ele, que é também equivalente a $||B|| - ||A||$. Por exemplo $dim([100, 101]) = 1 = ||101|| - ||100|| = 2 - 1$, onde $||A||$ representa o número de bits iguais a 1 em A.

2.4.5 Minimização Tabular de Quine-McCluskey

Definição 2.4.6 *Uma expressão Booleana é considerada uma* **expressão minimal** *se (1) não existe nenhuma outra expressão equivalente com um número menor de termos e (2) não existe nenhuma outra expressão equivalente com igual número de termos mas com menor número de literais.*

$\diamond$

A minimização de uma expressão Booleana pode ser realizada formalmente através de operações de simplificação de expressões da álgebra Booleana. Entretanto, a manipulação algébrica, além de ser uma tarefa cansativa, pode facilmente induzir uma pessoa a cometer erros, principalmente quando o número de variáveis envolvidas é grande.

O algoritmo tabular de Quine-McCluskey para minimização de funções Booleanas é um método clássico que sistematiza este processo algébrico. Embora existam outros algoritmos para minimização tais como o mapa de Karnaugh ([HP81], capítulo 6), o algoritmo de Quine-McCluskey é o melhor indicado para implementação em computadores digitais.

O algoritmo de Quine-McCluskey (QM) requer que as funções Booleanas estejam expressas nas formas canônicas. A idéia básica deste algoritmo consiste em encarar os mintermos da FND como pontos no n-espaço, ou seja, como vértices de um n-cubo. A partir do conjunto destes vértices (ou 0-cubos) procura-se gerar todos os 1-cubos possíveis combinando-se dois deles (equivale a gerar as arestas do cubo que ligam dois 0-cubos considerados). A partir da combinação de dois 1-cubos procura-se gerar todos os possíveis 2-cubos e assim por diante, até que nenhum cubo de dimensão maior possa ser gerado a partir da combinação de dois cubos de dimensão menor.

O processo de geração de k-cubos $(1 \leq k \leq n)$ a partir da combinação de dois $(k-1)$-cubos será denominado *passo* do algoritmo de minimização. Os cubos resultantes ao final de todo o processo são denominados **implicantes primos**.

Este processo de minimização pode ser facilmente associado ao processo algébrico de simplificação. Os mintermos da expressão na forma canônica inicial correspondem aos 0-cubos. Combinar dois 0-cubos para gerar um 1-cubo corresponde a combinar dois mintermos para eliminar uma variável e gerar um termo com menos literais para substituí-los, como mostramos no seguinte exemplo :

$$x_1x_2x_3 + x_1x_2\overline{x}_3 = x_1x_2(x_3 + \overline{x}_3) = x_1x_2 \cdot 1 = x_1x_2$$

Quando considerados no 3-espaço, o processo mostrado na expressão algébrica acima corresponde ao processo de agruparmos os 0-cubos 111 e 110 para geração do 1-cubo $11X$, como ilustra a figura 2.5.

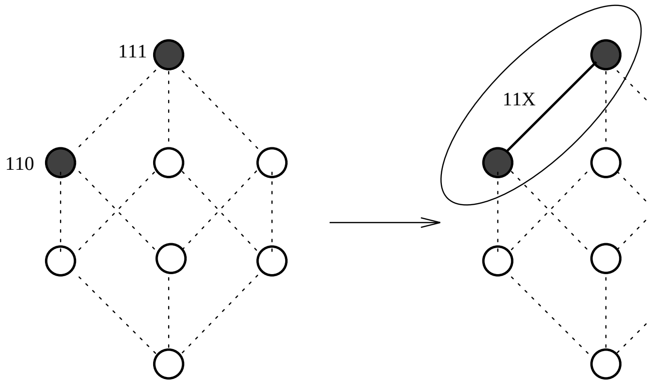


Figura 2.5: Passo elementar do algoritmo de Quine-McCluskey

A garantia de que o algoritmo QM realmente fornece uma expressão minimal, equivalente à expressão original, é uma decorrência óbvia do fato de que, em cada passo do algoritmo, são geradas todas as combinações possíveis entre os termos.

Descrição dos passos do algoritmo QM

A primeira parte do algoritmo QM consiste de um processo para determinação de todos os implicantes primos. A seguir descrevemos os processos que constituem esta parte e, ao mesmo tempo, mostramos a sua aplicação sobre a função $f(x_1, x_2, x_3) = \sum m(0, 1, 4, 5, 6)$.

- Primeiro passo : converter os mintermos para a notação binária.

  000, 001, 100, 101, 110

- Segundo passo : Separar os mintermos em grupos de acordo com o número de 1's em sua representação binária e ordená-los em ordem crescente, em uma coluna, separando os grupos com uma linha horizontal.

000
001
100
101
110

- Terceiro passo : combinar todos os elementos de um grupo com todos os elementos do grupo adjacente inferior para geração de cubos de dimensão maior. Para cada 2 grupos comparados entre si, gerar um novo grupo na próxima coluna e colocar os novos cubos. Marcar com $\surd$ os cubos que foram usados para gerar novos cubos.

$\surd$	000
$\surd$	001
$\surd$	100
$\surd$	101
$\surd$	110

$$\Longrightarrow$$

$00X$
$X00$
$X01$
$10X$
$1X0$

Observação : o novo cubo gerado será inserido no novo conjunto se e somente se ele ainda não estiver contido nele.

Repetir o processo para cada nova coluna formada, até que nenhuma combinação mais seja possível.

$\surd$	000
$\surd$	001
$\surd$	100
$\surd$	101
$\surd$	110

$$\Longrightarrow$$

$\surd$	$00X$
$\surd$	$X00$
$\surd$	$X01$
$\surd$	$10X$
	$1X0$

$$\Longrightarrow$$

$X0X$

- Quarto passo : Listar os implicantes primos. Os implicantes primos são aqueles que não foram combinados com nenhum outro, ou seja, aqueles que não estão com a marca $\surd$.

$$1X0 \text{ e } X0X$$

À primeira vista, poderíamos afirmar que a soma de todos os implicantes primos corresponde à expressão minimal da função Booleana. No entanto, existem casos em que a soma de dois ou mais implicantes primos cobre um outro. Neste caso, este último termo não é essencial, no sentido de que ele pode ser eliminado do conjunto de implicantes primos, sem que a expressão resultante deixe de ser equivalente à expressão original. Podemos ilustrar esta situação no seguinte exemplo .

Exemplo 2.4.5 *Considere a expressão Booleana* $f = 000 + 001 + 011 + 111$. *Pelo método de QM obtemos os seguintes implicantes primos :*

$$00X,\ 0X1 \text{ e } X11.$$

Graficamente, estes implicantes primos (ou cubos) correspondem respectivamente aos intervalos $[000, 001]$, $[001, 011]$ *e* $[011, 111]$ *ilustrados na figura 2.6a. Note, porém, que*

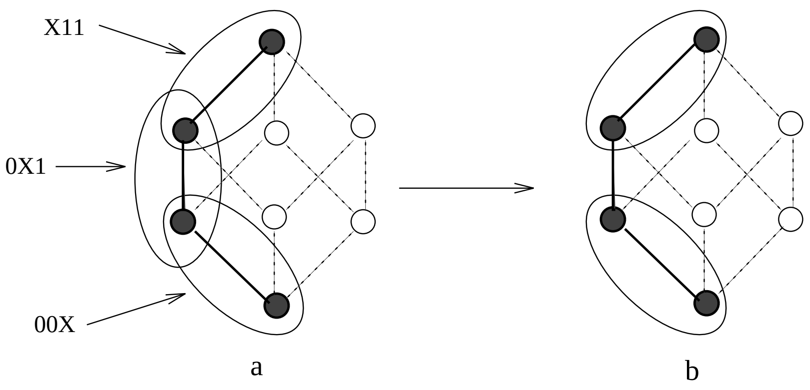


Figura 2.6: Implicantes primos (a) e implicantes primos essenciais (b).

o intervalo $[001, 011]$ *é supérfluo, ou seja, a mesma expressão pode ser expressa pelos implicantes primos* $00X$ *e* $X11$ *(figura 2.6b).*

◇

Portanto, o ponto central da segunda parte do algoritmo QM é o cálculo de um menor subconjunto do conjunto de implicantes primos suficientes para cobrir[2] todos os mintermos da função Booleana.

Definição 2.4.7 *Seja* $\mathcal{C}$ *uma coleção de intervalos maximais que cobre* X *e seja* $\mathcal{S} \subseteq \mathcal{C}$. *Então* $\mathcal{S}$ *é uma subcoleção ótima de* $\mathcal{C}$ *que cobre* X *se :*

a) não existe outra subcoleção $\mathcal{S}' \subseteq \mathcal{C}$ *que cobre* X, *tal que* $|\mathcal{S}'| < |\mathcal{S}|$;

b) $\forall \mathcal{S}' \subseteq \mathcal{C}$ *tal que* $\mathcal{S}'$ *cobre* X *e* $|\mathcal{S}'| = |\mathcal{S}|$, *não existe* $[A, B] \in \mathcal{S}'$ *tal que* $dim([A, B]) > dim([A', B'])$ *para todo* $[A', B'] \in \mathcal{S}$.

◇

Definição 2.4.8 *Uma subcoleção ótima dos implicantes primos que cobre todos os mintermos da expressão Booleana original é denominada conjunto dos* **implicantes primos essenciais**.

◇

[2]Um conjunto de implicantes primos (cubos maximais) cobre um mintermo (0-cubo) se este é coberto por pelo menos um dos implicantes primos.

Os implicantes primos essenciais são calculados com a ajuda de uma tabela denominada Tabela de Implicantes Primos ([HP81]), na qual é selecionado um subconjunto ótimo de implicantes primos que cobre todos os mintermos da expressão Booleana inicial.

A seguir descrevemos os passos da segunda parte do algoritmo QM e mostramos sua aplicação sobre a função $f(x_1, x_2, x_3, x_4, x_5) = \sum(1, 2, 3, 5, 9, 10, 11, 18, 19, 20, 21, 23, 25, 26, 27)$.

1. <u>Construir a Tabela de Implicantes Primos</u> : no topo das colunas deve-se colocar os mintermos de f e, à esquerda de cada linha, os implicantes primos. Os implicantes primos devem ser listados em ordem decrescente de acordo com a sua dimensão, isto é, em ordem crescente de acordo com o número de literais. Deve-se acrescentar uma coluna à esquerda e uma linha na parte inferior.

   Em cada linha, marcar as colunas com $\surd$ quando o implicante primo da linha cobre o mintermo da coluna.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
	XX01X		$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$					$\surd$	$\surd$
	X10X1					$\surd$		$\surd$						$\surd$		$\surd$
	0X0X1	$\surd$		$\surd$		$\surd$		$\surd$								
	00X01	$\surd$			$\surd$											
	X0101				$\surd$							$\surd$				
	1010X										$\surd$	$\surd$				
	10X11									$\surd$			$\surd$			
	101X1											$\surd$	$\surd$			

2. <u>Selecionar os implicantes primos essenciais</u> : deve-se procurar na tabela, as colunas que contém apenas uma marca $\surd$. A linha na qual estas colunas contém a marca $\surd$ corresponde a um implicante primo essencial. Em outras palavras, este implicante primo é o único que cobre o mintermo da coluna e, portanto, não pode ser descartado. Então, deve-se marcar com um asterisco (*) esta linha na coluna mais à esquerda, para indicar que este é um implicante primo essencial. A seguir, deve-se marcar, na última linha da tabela, todas as colunas cujo mintermo é coberto pelo implicante primo selecionado.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
*	XX01X		$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$					$\surd$	$\surd$
	X10X1					$\surd$		$\surd$						$\surd$		$\surd$
	0X0X1	$\surd$		$\surd$		$\surd$		$\surd$								
	00X01	$\surd$			$\surd$											
	X0101				$\surd$							$\surd$				
	1010X										$\surd$	$\surd$				
	10X11									$\surd$			$\surd$			
	101X1											$\surd$	$\surd$			
			$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$					$\surd$	$\surd$

   Deve-se repetir este processo enquanto existir uma coluna cujo mintermo não está coberto pelo conjunto de implicantes primos selecionados até o momento e que satisfaz as condições descritas.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
*	XX01X		√	√			√	√	√	√					√	√
*	X10X1					√		√						√		√
	0X0X1	√		√		√		√								
	00X01	√			√											
	X0101				√							√				
*	1010X										√	√				
	10X11									√			√			
	101X1											√	√			
			√	√		√	√	√	√	√	√	√		√	√	√

3. <u>Reduzir a tabela</u> : manter apenas as colunas correspondentes aos mintermos não cobertos pelos implicantes primos essenciais, e as linhas correspondentes aos implicantes primos que não foram selecionados e que cobrem pelo menos um mintermo ainda não coberto.

		1	5	23
	0X0X1	√		
	00X01	√	√	
	X0101		√	
	10X11			√
	101X1			√

4. <u>Selecionar os implicantes primos secundariamente essenciais</u> : eliminar as linhas dominadas e selecionar os essenciais, conforme definição a seguir:

Propriedade 2.4.1 *Diz-se que uma linha A na Tabela de Implicantes Primos domina uma outra linha B se e somente se $dim(A) \geq dim(B)$, e A cobre pelo menos todos os mintermos que estão nas colunas da tabela e que são cobertos por B. Se A domina B, então existe uma forma minimal que não utiliza o termo da linha B.* ◇

A seguir, deve-se repetir o mesmo processo do passo 2, porém os implicantes primos essenciais serão chamados secundariamente essenciais e marcados com dois asteriscos (**), e em seguida fazer a redução da tabela (passo 3).

		1	5	23
	0X0X1	√		
**	00X01	√	√	
**	10X11			√
		√	√	√

Deve-se repetir este processo até que isto não seja mais possível, ou seja, até atingir-se um estágio em que não mais existe, na tabela reduzida, nenhuma coluna com apenas um √ e não existe nenhuma linha que cobre outra.

<u>Aplicar o método de Petrick</u> : este método (de busca exaustiva) fornece todas as possíveis combinações dos implicantes primos restantes que cobrem os mintermos restantes. Deve-se escolher dentre elas, aquela que envolve o menor número de termos. Caso existam mais de uma nestas condições, deve-se escolher a de custo mínimo (aquela que envolve menor número de literais).

O algoritmo QM

Apresentamos a seguir o algoritmo QM em pseudo código :

```
Entrada :  Conjunto de mintermos
Saída :    Conjunto de implicantes primos essenciais

1) Separe os mintermos em n blocos B0_i, i = 0,1,2,3,...,n
           fazendo B0_i = { todos os mintermos com i 1's };
2) Para i de 0 até n − 1 faça
           Combine cada mintermo de B0_i com os mintermos de B0_{i+1} e
              se estes formam um 1-cubo, então
                 - marque estes dois mintermos;
                 - coloque o 1-cubo no bloco B1_i;
3) Coloque todos os mintermos que não foram marcados na lista
              de implicantes primos;
4) Repita o mesmo processo para os blocos B1_i e B1_{i+1} para
              i variando de 0 até n − 2 e coloque os 2-cubos nos blocos B2_i;
5) Coloque todos os 1-cubos que não foram marcados na
              lista de implicantes primos;
6) Repita este processo sucessivamente para gerar 3-cubos, 4-cubos, etc,
              até que nenhuma combinação seja mais possível;
7) Construa a Tabela de implicantes primos e selecione
              os implicantes primos essenciais;
```

O efeitos do algoritmo QM sobre a expressão $f = \sum(0,1,4,5,6)$ podem ser graficamente vizualizados na figura 2.7. A função minimal correspondente é $f' = \overline{x}_2 + x_1\overline{x}_3$.

Complexidade do Algoritmo QM

A complexidade[3] de tempo e espaço deste algoritmo no pior caso é exponencial, uma vez que o número máximo de mintermos possíveis é $\mathcal{O}(2^n)$. Além disso, sabe-se que o problema de escolha dos implicantes primos essenciais é um problema $\mathcal{NP}$-completo ([GJ79]).

[3] O termo complexidade é utilizado para caracterizar a "eficiência" de um algoritmo. Informalmente, dizer que um algoritmo tem complexidade de tempo exponencial significa dizer que seu tempo de processamento cresce em proporção exponencial em relação ao crescimento do tamanho que caracteriza os dados de entrada. Uma definição formal pode ser encontrada por exemplo em [TLR90]

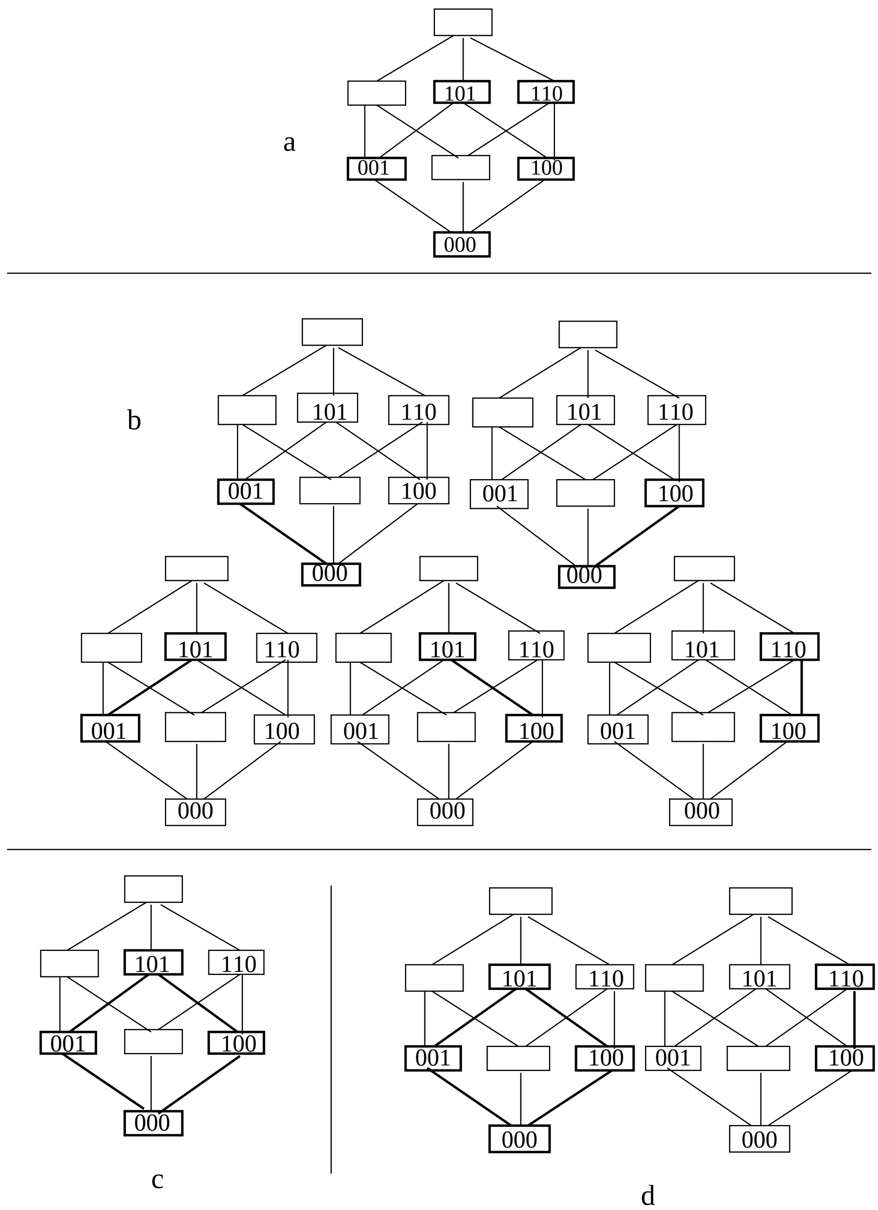


Figura 2.7: Visualização do efeito causado pelo algoritmo de Quine-McCluskey sobre a função $f = \sum(0,1,4,5,6)$. a) os 0-cubos que inicializam o algoritmo. b) os 1-cubos obtidos combinado-se os 0-cubos. c) os 2-cubos obtidos combinado-se os 1-cubos. d) os implicantes primos $X0X$ e $1X0$ ($\overline{x}_2$ e $x_1\overline{x}_3$).

2.5 Outros Conceitos Básicos

Grupo Abeliano

Um conjunto G, munido com uma operação binária $+$, é um **grupo** se satisfaz os seguintes axiomas :

- $a + (b + c) = (a + b) + c, \quad \forall a, b, c \in G,$
- G contém um elemento e tal que $e + a = a + e = a$ para cada $a \in G$,
- para cada $a \in G$, existe um elemento em G, denotado por $-a$, tal que $a + (-a) = (-a) + a = e$.

Se o grupo G satisfaz $a + b = b + a$, para qualquer $a, b \in G$, então G é denominado **grupo comutativo** ou **grupo Abeliano** ([MSM63]).

Capítulo 3

Morfologia Matemática

Os problemas em Análise de Imagens, quando restritos ao domínio das imagens binárias, podem ser modelados por transformações entre subconjuntos. Conseqüentemente, o estudo da *MM* para subconjuntos pode fornecer as soluções para estes problemas.

Para justificar esta afirmativa, mostraremos inicialmente que existe uma relação biunívoca entre as imagens binárias definidas em um determinado domínio e os subconjuntos deste mesmo domínio.

Em seguida, apresentaremos os elementos da *MM* necessários para enunciar os teoremas de decomposição canônica e minimal, de uma particular classe de operadores denominados invariantes por translação e localmente definidos. Os vários conjuntos criados a partir das noções definidas no estudo destas decomposições serão relacionados através da investigação das relações de isomorfismo existentes entre eles.

Antes de prosseguirmos, lembramos que a *MM* estuda mapeamentos entre dois reticulados quaisquer. Entretanto, devido ao nosso interesse em imagens binárias, os conceitos e resultados apresentados serão restritos ao reticulado Booleano das partes de um conjunto.

3.1 Imagens Binárias e sua Equivalência com Subconjuntos

Neste texto, vamos supor que as imagens encontram-se definidas sobre um subconjunto não vazio $E \subseteq \mathbb{Z}^2$, isto é, $E = E_1 \times E_2$, onde $E_1, E_2 \subseteq \mathbb{Z}$.

Uma imagem definida sobre E pode ser representada por uma função f que a cada ponto de E associa um valor entre 0 e $k-1$, chamado nível de cinza, isto é, $f : E \to \{0, 1, ..., k-1\}$. O valor k corresponde ao número de tons de cinza presentes na imagem.

Em particular, as **imagens binárias** são representadas por funções do tipo $f : E \to \{0,1\}$.

Denotaremos por $\{0,1\}^E$ o conjunto de todas as funções definidas de E em $\{0,1\}$, que podem ser interpretadas como o conjunto de todas as imagens binárias definidas em E.

Proposição 3.1.1 *Sejam $x \in E$ e $X \subseteq E$. A função definida por*

$$\mathbf{1}_X(x) = \begin{cases} 1, & se\ x \in X \\ 0, & caso\ contr\acute{a}rio \end{cases}$$

é uma bijeção de $\mathcal{P}(E)$ em $\{0,1\}^E$ e sua inversa é dada por

$$suporte(f) = \{x \in E : f(x) \neq 0\}.$$

Dem.: A demostração pode ser encontrada em [BB94], página 8. ◇

A proposição acima garante que existe uma relação de 1 para 1 entre $\{0,1\}^E$ e $\mathcal{P}(E)$. Portanto, uma imagem binária definida em E pode ser representada por um subconjunto $X \subseteq E$.

Exemplo 3.1.1 *A figura 3.1 ilustra todos os subconjuntos de um quadrado 2×2 (Os quadriculados escuros correspondem aos pontos que pertencem ao subconjunto). Cada um dos subconjuntos pode ser interpretado como uma imagem binária definida numa grade de dimensão 2×2. A coleção dos quadriculados escuros e brancos correspondem, respectivamente, à figura e ao fundo da imagem.*

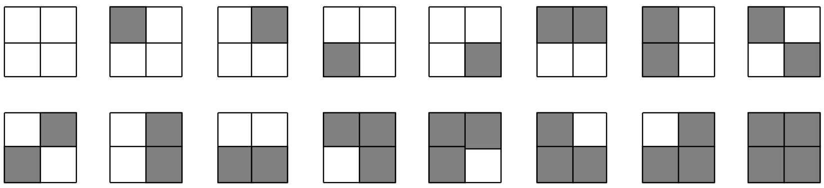


Figura 3.1: Todos os subconjuntos de um quadrado 2×2.

◇

3.2 Decomposição de Operadores

Uma função $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ será denominada **operador morfológico binário**, ou simplesmente **operador**, e pode ser interpretada como uma transformação de imagens binárias.

Vamos supor que o domínio E munido com uma operação de adição, denotada por $+$, forma um grupo abeliano $(E, +)$ (veja [BB94], capítulo 2). O elemento neutro de E, também denominado origem, será denotado por o e o oposto de $x \in E$ será denotado por $-x$.

A seguir apresentamos algumas definições necessárias mais adiante.

Definição 3.2.1 *Sejam $X \in \mathcal{P}(E)$ e $h \in E$. Então,*

$$X + h = \{x + h : x \in X\}$$

é definido como o **translado de X por h**. ◇

Definição 3.2.2 *Seja $X \subseteq E$. O* **transposto de X** *(em relação a origem) é o conjunto*

$$X^t = \{x \in E : -x \in X\}.$$

◇

Definição 3.2.3 *Um operador $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ é dito* **invariante por translação***, ou simplesmente* **i.t.***, sse*

$$\psi(X + h) = \psi(X) + h \quad (X \in \mathcal{P}(E),\ h \in E).$$

◇

Definição 3.2.4 *Seja $W \subseteq E$, W finito. Um operador $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ é* **localmente definido pela janela W** *sse*

$$x \in \psi(X \cap (W + x)) \Longleftrightarrow x \in \psi(X)$$

para todo $x \in E$ e $X \in \mathcal{P}(E)$. ◇

Os primeiros resultados sobre decomposição de operadores foram introduzidos por G. Matheron em 1975. O Teorema de Representação de Matheron ([Mat75]) estabelece que qualquer operador i.t. e crescente[1] pode ser expresso como uma união de operadores elementares denominados erosões, ou dualmente, como uma interseção de operadores elementares denominados dilatações. Uma generalização destes resultados para operadores i.t., mas não necessariamente crescentes, foi introduzida por Banon e Barrera em [BB91].

Nesta seção apresentaremos resultados referentes à decomposição de operadores i.t., localmente definidos por uma janela W. Em relação aos resultados de Banon e Barrera, no qual a noção de operadores localmente definidos por uma janela não é levada em consideração, os resultados a serem apresentados aqui têm um caráter mais geral, no sentido que aqueles são um caso particular deste, quando consideramos a janela $W = E$.

[1] $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ é crescente se $X \subseteq Y \Longrightarrow \psi(X) \subseteq \psi(Y),\ \forall X, Y \in \mathcal{P}(E)$.

3.2.1 Operadores Elementares

No capítulo 1, comentamos que um operador pode ser expresso em termos de quatro operadores elementares (erosão, anti-erosão, dilatação, anti-dilatação) e as operações de supremo e ínfimo. No entanto, quando o reticulado possui uma operação de negação, o conjunto consistindo dos operadores erosão e dilatação e as operações de negação, ínfimo e supremo é suficiente para representar um operador qualquer. No caso do reticulado das partes de um conjunto, a negação corresponde ao operador complemento e as operações de ínfimo e supremo correspondem, respectivamente, à interseção e à união de conjuntos.

A seguir definiremos os elementos necessários para a apresentação do Teorema da Decomposição Canônica.

Definição 3.2.5 *Sejam $X, B \subseteq E$. A* **erosão** *de X por B é definida por :*

$$\varepsilon_B(X) = \{x \in E : (B + x) \subseteq X\}.$$

$\diamond$

Exemplo 3.2.1 *A figura 3.2 mostra uma aplicação de um operador erosão. O sinal + indica a origem de B.*

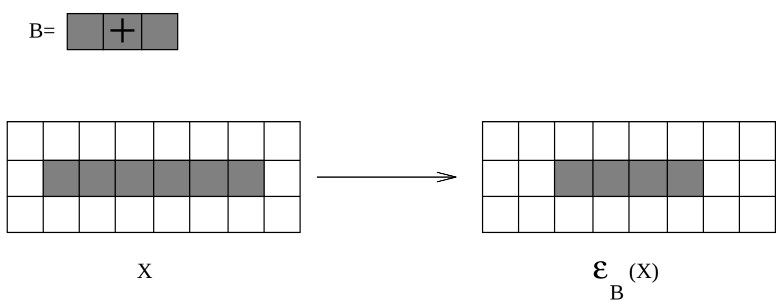


Figura 3.2: Erosão de X por B.

Definição 3.2.6 *Sejam $X, B \subseteq E$. A* **dilatação** *de X por B é definida por :*

$$\delta_B(X) = \{x \in E : (B^t + x) \cap X \neq \emptyset\}.$$

$\diamond$

Exemplo 3.2.2 *A figura 3.3 mostra uma aplicação de um operador dilatação.* $\diamond$

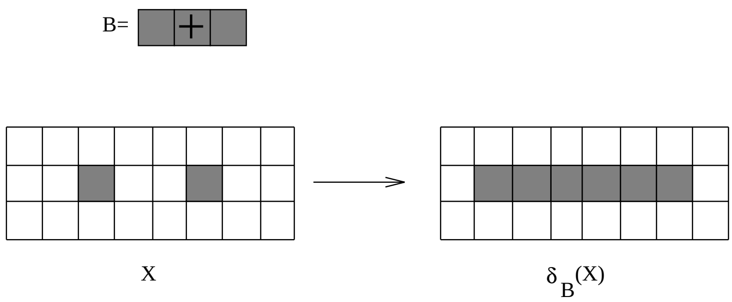


Figura 3.3: Dilatação de X por B.

Definição 3.2.7 *Seja $W \subseteq E$, W finito, e sejam $A, B \in \mathcal{P}(W)$, $A \subseteq B$. O operador definido por*

$$\lambda^W_{(A,B)}(X) = \{x \in E : A + x \subseteq X \cap (W + x) \subseteq B + x\} \quad (X \in \mathcal{P}(E))$$

é denominado **sup-gerador**. ◇

Exemplo 3.2.3 *A figura 3.4 ilustra o uso de um operador sup-gerador para extração de pontos extremos esquerdos de linhas horizontais de espessura 1 pixel em imagens binárias.*

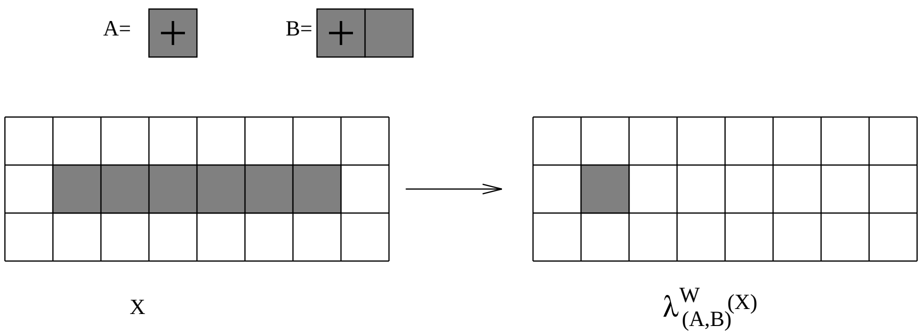


Figura 3.4: Extração de ponto extremo esquerdo através de um operador sup-gerador.

◇

O conjunto $B \subseteq E$ que parametriza, respectivamente nas definições 3.2.5 e 3.2.6, a erosão (ε_B) e a dilatação (δ_B) é denominado **elemento estruturante**. A seguinte proposição estabelece uma relação entre os operadores definidos acima.

Proposição 3.2.1 *Um operador sup-gerador $\lambda^W_{(A,B)}$ pode ser expresso como a interseção entre uma erosão e o complemento de uma dilatação ([BB91]), caracterizados respectivamente pelos elementos estruturantes A e B^{ct}, isto é,*

$$\lambda^W_{(A,B)}(X) = \varepsilon_A(X) \cap (\delta_{B^{ct}}(X))^c \quad (X \in \mathcal{P}(E)).$$

◇

Observe que os operadores acima definidos são todos invariantes por translação e localmente definidos.

Exemplo 3.2.4 *A figura 3.5 ilustra a decomposição da sup-geradora ilustrada na figura 3.4 em termos de uma erosão e uma anti-dilatação (complemento de uma dilatação).*

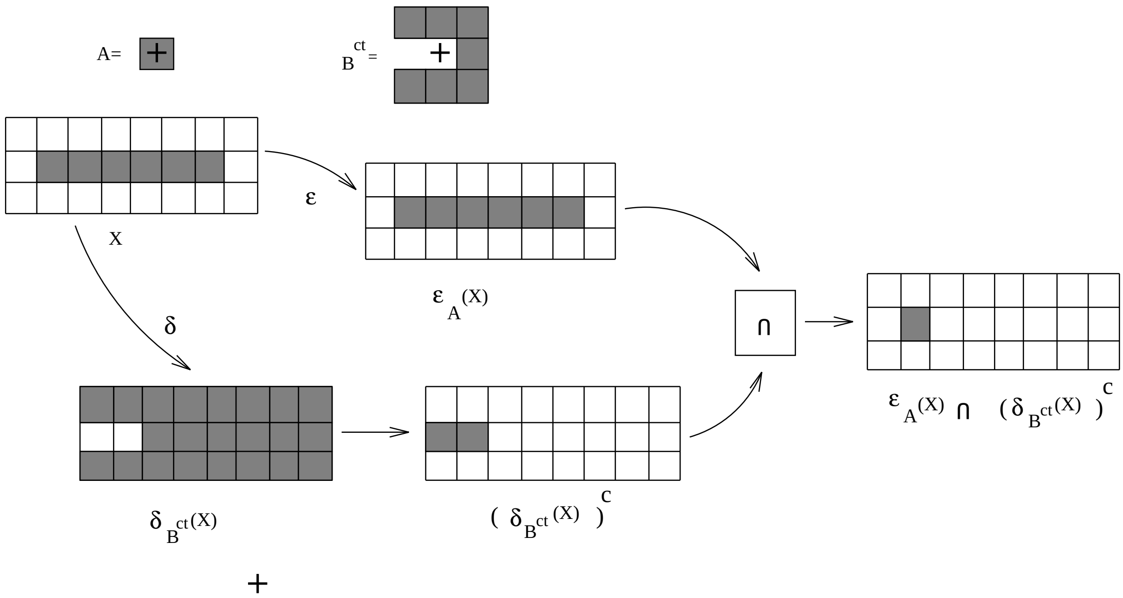


Figura 3.5: Erosão e dilatação que compõem uma sup-geradora.

◇

3.2.2 Decomposição Canônica

Um importante conceito que está associado à decomposição canônica de um operador é o seu núcleo ([BS95]), cuja noção apresentaremos na seguinte definição.

Definição 3.2.8 *O* **núcleo**, *ou* **kernel**, *de um operador $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$, i.t. e localmente definido por uma janela W, é definido por*

$$\mathcal{K}_W(\psi) = \{Y \in \mathcal{P}(W) : o \in \psi(Y)\}.$$

Teorema 3.2.1 (Decomposição canônica [BB91]) [2] *Seja $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ um operador i.t. localmente definido por uma janela W. Então,*

$$\psi(X) = \bigcup\{\lambda^W_{(A,B)}(X) : [A,B] \subseteq \mathcal{K}_W(\psi)\} \quad (X \in \mathcal{P}(E)).$$

◇

O teorema 3.2.1 garante que qualquer operador i.t. e localmente definido por uma janela W pode ser escrito como a união de um conjunto de operadores sup-geradores que, por sua vez, podem ser expressos como a interseção entre uma erosão e o complemento de uma dilatação (Proposição 3.2.1). Portanto, qualquer operador i.t. pode ser expresso em termos de erosões, dilatações, complemento, interseção e união. Mais ainda, o teorema explicita os elementos estruturantes (A e B) que caracterizam, respectivamente, as erosões e as dilatações.

Exemplo 3.2.5 *A figura 3.6 ilustra a decomposição canônica do operador que extrai pontos extremos de segmentos de retas horizontais, com espessura de 1 pixel. Observe que este operador é a união de dois operadores sup-geradores, um para extração de pontos extremos à esquerda e outro para extração de pontos extremos à direita.*

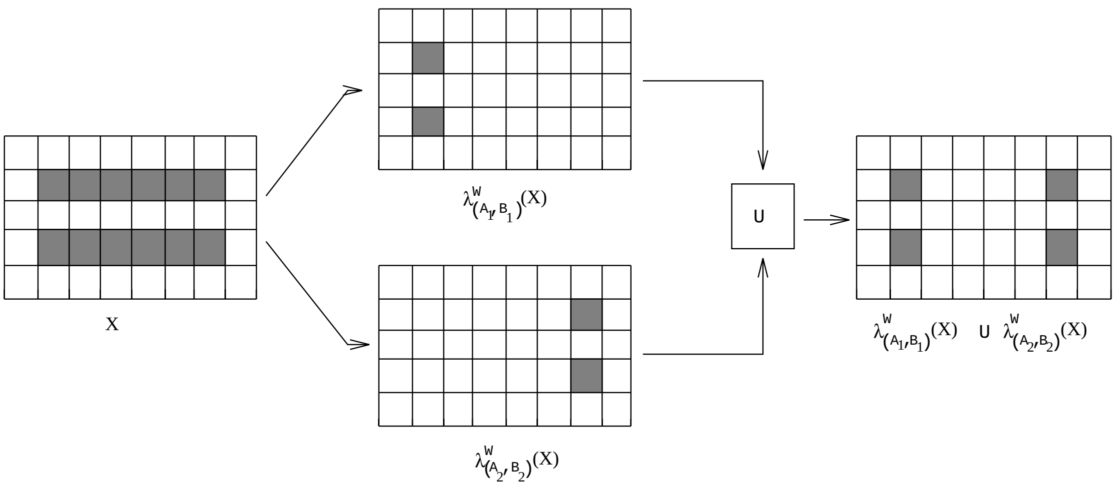


Figura 3.6: Decomposição canônica

◇

3.2.3 Decomposição Minimal

Os primeiros resultados de decomposição minimal foram apresentados por Maragos ([Mar85]) para a decomposição apresentada por Matheron. Seguindo a mesma idéia, Banon e Barrera introduziram o Teorema de Decomposição Minimal ([BB91]) para a decomposição canônica.

Não há dúvida de que, sob o aspecto teórico, o teorema da decomposição canônica representa um resultado bastante interessante. No entanto, do ponto de vista prático, uma simples análise mostra que existem redundâncias nesta decomposição. Basta verificarmos que, se $[A, B]$ e $[A', B']$ são dois intervalos contidos em $\mathcal{K}_W(\psi)$ tais que $[A, B] \subseteq [A', B']$, então, pela definição 3.2.7, $\forall X \in \mathcal{P}(E)$,

$$\lambda^W_{(A,B)}(X) \subseteq \lambda^W_{(A',B')}(X)$$

e, portanto, $\lambda^W_{(A,B)}(X)$ é um elemento redundante pois :

$$\lambda^W_{(A,B)}(X) \subseteq \lambda^W_{(A',B')}(X) \Longrightarrow \lambda^W_{(A,B)}(X) \cup \lambda^W_{(A',B')}(X) = \lambda^W_{(A',B')}(X).$$

Baseado na observação acima, podemos definir um outro conjunto equivalente ao núcleo, porém de menor cardinalidade, que também caracteriza a decomposição.

Definição 3.2.9 *Seja $\mathcal{X}$ uma coleção de intervalos, isto é, $\mathcal{X} = \{[A_i, B_i] \subseteq \mathcal{P}(W) : i \in I\}$, onde I é um conjunto de índices. O intervalo $[A, B] \in \mathcal{X}$ é* **maximal** *em $\mathcal{X}$ se e somente se $\nexists [A', B'] \in \mathcal{X}$, tal que $[A, B] \subset [A', B']$ (i.e., $[A, B] \subseteq [A', B']$) e $[A', B'] \neq [A, B]$).*

$\diamond$

Definição 3.2.10 *Seja $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ um operador i.t. e localmente definido por uma janela W. O conjunto $\mathcal{B}_W(\psi)$ de todos os intervalos maximais de $\mathcal{K}_W(\psi)$ é denominado* **base** *de ψ.*

$\diamond$

Teorema 3.2.2 (Decomposição Minimal) *Seja $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ um operador i.t. e localmente definido por uma janela W e seja $\mathcal{B}_W(\psi)$ a sua base . Então,*

$$\psi(X) = \bigcup \{\lambda^W_{(A,B)}(X) : [A, B] \in \mathcal{B}_W(\psi)\} \quad (X \in \mathcal{P}(E)).$$

$\diamond$

Os resultados que foram apresentados nesta seção possuem uma versão dual. Os operadores duais às sup-geradoras são as inf-geradoras, definidos como a união entre uma dilatação e uma anti-erosão (complemento de uma erosão). Os duais correspondentes aos teoremas 3.2.1 e 3.2.2 são aqueles nos quais os operadores são escritos como interseção de operadores inf-geradores. Não aprofundaremos as explanações sobre os resultados duais uma vez que eles podem ser derivados facilmente a partir dos resultados apresentados.

3.3 Operadores Vistos Através de Isomorfismos

Na seção anterior vimos um importante resultado sobre os operadores invariantes por translação e localmente definidos por uma janela. Dentro deste contexto de decomposição canônica de operadores, podemos considerar os seguintes conjuntos :

- o conjunto Ψ_W de todos os operadores i.t. localmente definidos por W,
- o conjunto $\mathcal{P}(\mathcal{P}(W))$ de coleções de subconjuntos de W (que podem ser interpretados como todos os possíveis núcleos) e,
- o conjunto $\mathcal{M}(W)$ de coleções de intervalos maximais (que podem ser interpretados como todas as possíveis bases).

Nesta seção iremos estudar a estrutura destes conjuntos e investigar a relação existente entre eles.

3.3.1 O Conjunto Ψ_W

Seja Ψ o conjunto de todos os mapeamentos de $\mathcal{P}(E)$ em $\mathcal{P}(E)$. O conjunto Ψ herda a estrutura de reticulado Booleano do conjunto $(\mathcal{P}(E), \subseteq)$, quando definimos a relação de ordem $\preceq$ dada por :

$$\psi_1 \preceq \psi_2 \iff \psi_1(X) \subseteq \psi_2(X), \quad (\psi_1, \psi_2 \in \Psi \text{ e } X \in \mathcal{P}(E)).$$

O ínfimo ($\bigwedge$) e o supremo ($\bigvee$) de uma família de operadores $\Theta \subseteq \Psi_W$, e o complemento de um operador $\psi \in \Psi_W$ são dados, respectivamente, por :

$$(\bigwedge \Theta)(X) = \bigcap\{\theta(X) : \theta \in \Theta\}, \; X \in \mathcal{P}(E),$$

$$(\bigvee \Theta)(X) = \bigcup\{\theta(X) : \theta \in \Theta\}, \; X \in \mathcal{P}(E)$$

e

$$(\nu\psi)(X) = (\psi(X))^c.$$

O conjunto Ψ_W, com a relação de ordem induzida por $\preceq$, constitui um subreticulado Booleano de Ψ. Usaremos a mesma notação , $\preceq$, para a relação de ordem induzida por Ψ sobre Ψ_W.

3.3.2 O Conjunto $\mathcal{P}(\mathcal{P}(W))$

É fácil verificarmos que o conjunto $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$ onde $\subseteq$ é definido por $\mathcal{X} \subseteq \mathcal{Y} \Longleftrightarrow X \in \mathcal{X} \to X \in \mathcal{Y}$, $\forall \mathcal{X}, \mathcal{Y} \in \mathcal{P}(\mathcal{P}(W))$, e o supremo e o ínfimo de uma coleção $\{\mathcal{X}_i : \mathcal{X}_i \in \mathcal{P}(\mathcal{P}(W)), i \in I\}$ e o complemento de $\mathcal{X} \in \mathcal{P}(\mathcal{P}(W))$ são definidos, respectivamente, por

$$\bigvee \mathcal{X}_i = \{X \in \mathcal{P}(W) : X \in \mathcal{X}_i \text{ para algum } i \in I\},$$

$$\bigwedge \mathcal{X}_i = \{X \in \mathcal{P}(W) : X \in \mathcal{X}_i, \ \forall i \in I\}$$

e

$$\mathcal{X}^c = \mathcal{P}(W) - \mathcal{X} = \{X \in \mathcal{P}(W) : X \notin \mathcal{X}\}$$

é um reticulado Booleano completo.

3.3.3 O Conjunto $\mathcal{M}(W)$

Seja $\mathcal{M}(W)$ o conjunto de todas as coleções de intervalos maximais contidos em $\mathcal{P}(W)$, isto é,

$$\mathcal{M}(W) = \{Max(\mathcal{X}), \ \mathcal{X} \subseteq \mathcal{P}(W)\}$$

onde $Max(\mathcal{X})$ denota o conjunto de intervalos maximais da coleção $\{[A, B] \subseteq \mathcal{P}(W) : [A, B] \subseteq \mathcal{X}\}$.

Para $\boldsymbol{X} \in \mathcal{M}(W)$, denotaremos

$$\bigcup \boldsymbol{X} = \{Y \in \mathcal{P}(W) : \exists \ [A, B] \in \boldsymbol{X}, \ Y \in [A, B]\}.$$

Seja $\mathcal{X} \subseteq \mathcal{P}(W)$. Observe que $\bigcup Max(\mathcal{X}) = \mathcal{X}$. Logo, $\mathcal{X}$ pode ser representado pelo conjunto de intervalos maximais $Max(\mathcal{X})$, o qual denotaremos, por simplicidade, também por $\boldsymbol{X}$.

Definimos uma relação binária $\sqsubseteq$ em $\mathcal{M}(W)$ da seguinte forma :

$$\forall \ \boldsymbol{X}, \boldsymbol{Y} \in \mathcal{M}(W), \ \boldsymbol{X} \sqsubseteq \boldsymbol{Y} \Longleftrightarrow \forall \ [A, B] \in \boldsymbol{X}, \exists \ [A', B'] \in \boldsymbol{Y} : [A, B] \subseteq [A', B'].$$

A relação $\sqsubseteq$ é uma relação de ordem parcial.

O conjunto $\mathcal{M}(W)$ com a relação de ordem $\sqsubseteq$ definida acima é um reticulado Booleano completo, onde o supremo ($\sqcup$), o ínfimo ($\sqcap$) e o complemento são definidos, respectivamente, por :

$$\boldsymbol{X}_1 \sqcup \boldsymbol{X}_2 = Max(\mathcal{X}_1 \cup \mathcal{X}_2), \ \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathcal{M}(W)$$

$$\boldsymbol{X}_1 \sqcap \boldsymbol{X}_2 = Max(\mathcal{X}_1 \cap \mathcal{X}_2), \ \boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathcal{M}(W)$$

$$\overline{\boldsymbol{X}} = Max(\mathcal{X}^c), \ X \in \mathcal{M}(W).$$

3.3.4 Relações de Isomorfismo

Pode-se verificar facilmente que o conjunto das funções Booleanas, $(\{0,1\}^{\mathcal{P}(W)}, \leq)$, também é um reticulado Booleano, onde $\leq$ é definido por $f_1 \leq f_2 \iff f_1(X) \leq f_2(X)$, $\forall X \in \mathcal{P}(W)$.

Portanto, temos em mãos quatro estruturas de reticulado completo Booleano. Uma investigação mais profunda sobre possíveis relações entre eles mostra-nos a existência de isomorfismo de reticulados entre eles, dos quais alguns são indicados na figura 3.7 (Veja [BS95]). As seguintes quatro proposições são dedicadas para especificar cada uma dessas funções.

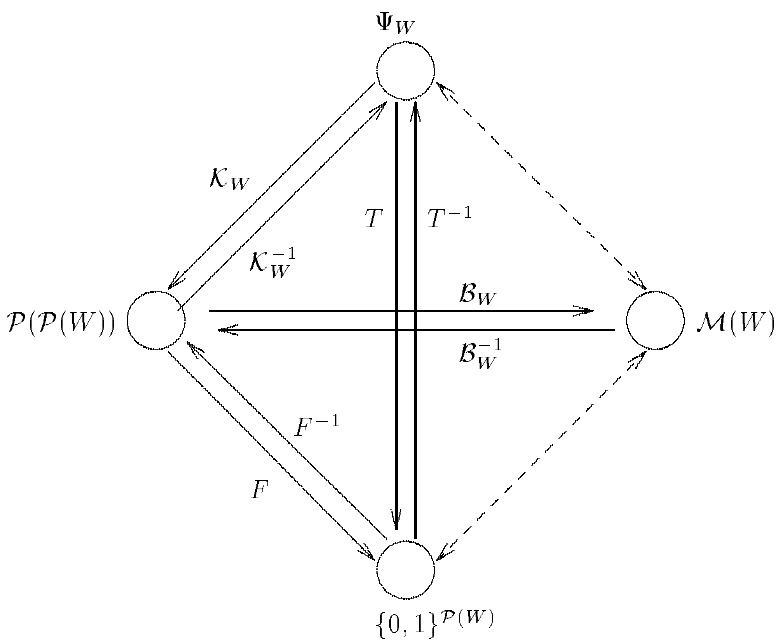


Figura 3.7: Diagrama dos isomorfismos.

Proposição 3.3.1 *O mapeamento $\mathcal{K}_W(\psi)$ de Ψ_W em $\mathcal{P}(\mathcal{P}(W))$ definido por*

$$\mathcal{K}_W(\psi) = \{X \in \mathcal{P}(W) : o \in \psi(X)\}, \quad (\psi \in \Psi_W)$$

constitui um isomorfismo de reticulados entre $(\Psi_W, \preceq)$ e $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$. A sua inversa é definida por

$$\mathcal{K}_W^{-1}(\mathcal{X})(X) = \{x \in E : (X - x) \cap W \in \mathcal{X}\}, \quad X \in \mathcal{P}(W).$$

$\diamond$

Proposição 3.3.2 *Seja F uma função definida de $\mathcal{P}(\mathcal{P}(W))$ em $\{0,1\}^{\mathcal{P}(W)}$ por*

$$F(\mathcal{X})(X) = \begin{cases} 1 & \textit{se } X \in \mathcal{X} \quad (X \in \mathcal{P}(W)) \\ 0 & \textit{caso contrário} \end{cases}$$

e

$$F^{-1}(f) = \{X \in \mathcal{P}(W) : f(X) = 1\} \quad (f \in \{0,1\}^{\mathcal{P}(W)}).$$

Então, F é um isomorfismo de reticulados entre $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$ e $(\{0,1\}^{\mathcal{P}(W)}, \leq)$. $\diamond$

Proposição 3.3.3 *Seja T uma função de Ψ_W em $\{0,1\}^{\mathcal{P}(W)}$ definida por*

$$T(\psi)(X) = \begin{cases} 1, & \textit{se } o \in \psi(X) \quad (X \in \mathcal{P}(W)) \\ 0, & \textit{caso contrário.} \end{cases}$$

e

$$T^{-1}(f)(X) = \{x \in E : f((X - x) \cap W) = 1\} \quad (X \in \mathcal{P}(W)).$$

Então, T é um isomorfismo entre $(\Psi_W, \preceq)$ e $(\{0,1\}^{\mathcal{P}(W)}, \leq)$. $\diamond$

Vamos denotar por ψ_f o operador de Ψ_W isomorfo a $f \in \{0,1\}^{\mathcal{P}(W)}$ e por f_ψ a função Booleana isomorfa a $\psi \in \Psi_W$.

Proposição 3.3.4 *Seja $\mathcal{B}_W$ a função definida de $\mathcal{P}(\mathcal{P}(W))$ em $\mathcal{M}(W)$ por :*

$$\mathcal{B}_W(\mathcal{X}) = Max(\mathcal{X}), \quad \forall \mathcal{X} \in \mathcal{P}(\mathcal{P}(W))$$

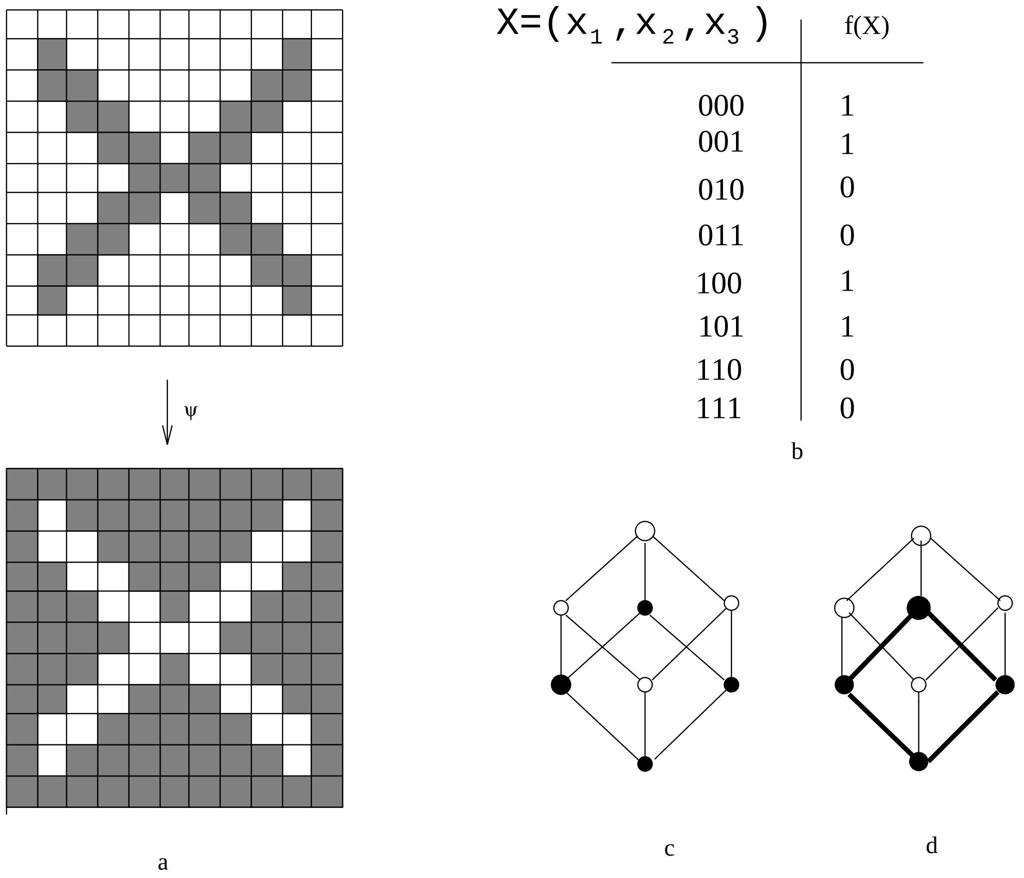
e

$$\mathcal{B}_W^{-1}(\boldsymbol{X}) = \{Y \in \mathcal{P}(W) : Y \in [A,B], [A,B] \in \boldsymbol{X}\}, \quad \forall \boldsymbol{X} \in \mathcal{M}(W).$$

Então, $\mathcal{B}_W$ é um isomorfismo entre os reticulados $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$ e $(\mathcal{M}(W), \sqsubseteq)$. $\diamond$

Observe que, a partir destas relações, podemos concluir que os quatro reticulados ilustrados na figura 3.7 são isomorfos 2 a 2.

Exemplo 3.3.1 *Na figura 3.8 ilustramos as diversas maneiras de representar o operador que calcula o complemento de uma imagem. A figura 3.8a mostra as imagens anterior e posterior à transformação. O núcleo deste operador é mostrado através de elementos do reticulado (círculos pretos) na figura 3.8c e a sua base (círculos pretos e arestas em negrito) na figura 3.8d. A função Booleana correspondente é definida pela tabela-verdade da figura 3.8b.*



$X=(x_1,x_2,x_3)$	f(X)
000	1
001	1
010	0
011	0
100	1
101	1
110	0
111	0

Figura 3.8: Equivalência de elementos dos quatro reticulados Booleanos.

3.4 Algumas Conseqüências Interessantes

Sob a óptica dos isomorfismos estudados, é possível apresentarmos uma prova elegante para o Teorema da Decomposição Canônica. Se observarmos que

$$\mathcal{K}_W(\psi) = \bigcup\{[A, B] : [A, B] \subseteq \mathcal{K}_W(\psi)\}$$

e

$$\mathcal{K}(\lambda^W_{(A,B)}) = [A, B],$$

temos que

$$\mathcal{K}_W(\psi) = \bigcup\{\mathcal{K}(\lambda^W_{(A,B)}) : [A, B] \subseteq \mathcal{K}_W(\psi)\}.$$

Tomando a inversa de $\mathcal{K}_W$, temos

$$\mathcal{K}_W^{-1}\mathcal{K}_W(\psi) = \bigvee\{\mathcal{K}_W^{-1}\mathcal{K}(\lambda^W_{(A,B)}) : [A, B] \subseteq \mathcal{K}_W(\psi)\}$$

de onde decorre que

$$\psi = \bigvee\{\lambda^W_{(A,B)} : [A, B] \subseteq \mathcal{K}_W(\psi)\}.$$

O fato destes 4 reticulados completos Booleanos serem isomorfos 2 a 2 permite que um determinado problema definido em um destes espaços possa ser estudado em qualquer um dos outros. Em outras palavras, o estudo do espaço dos operadores em busca de um operador para resolver um determinado problema, pode ser equivalentemente substituído pelo estudo do espaço dos núcleos, das bases ou das funções Booleanas. Neste sentido, um resultado teórico em um espaço possui um equivalente nos outros espaços; por exemplo, as formas canônicas dos operadores correspondem às formas normais das funções Booleanas.

3.4.1 Relação entre Operadores e Funções Booleanas

Verificamos que existe um isomorfismo de reticulados entre o conjunto Ψ_W e o conjunto $\{0,1\}^{\mathcal{P}(W)}$, a partir do qual não é difícil concluirmos que a forma canônica e minimal dos operadores correspondem, respectivamente, à forma canônica e minimal das funções Booleanas.

Uma regra para converter expressão Booleana em operador

Sejam $x_1, x_2, ..., x_n$ e $\overline{x}_1, \overline{x}_2, ..., \overline{x}_n$ os literais de uma função Booleana f de n variáveis, expressas como soma de produtos, e seja W a janela associada a estas n variáveis. Um produto m_j de f é composto de no máximo n literais, tais que se x_i está presente em m_j, então $\overline{x}_i$ não está presente e vice-versa. Dado um produto m_j, então o correspondente intervalo $[A, B]$, que caracteriza a sup-geradora na decomposição do operador correspondente ψ_f é obtido da seguinte forma :

$$A = \{\text{todos os pontos de } W \text{ correspondentes aos literais não barrados em } m_j\}$$

e

$$B = \left\{ \begin{array}{c} \text{todos os pontos de } W \text{ menos os pontos} \\ \text{correspondentes aos literais barrados em } m_j \end{array} \right\}$$

Exemplo 3.4.1 *A função Booleana constante igual a* 1 *corresponde ao operador cuja base é* $\{[\emptyset, W]\}$ *e a função Booleana constante igual a* 0 *corresponde ao operador cuja base é* $\emptyset$*. Se* $W = (x_1, x_2, x_3)$*, os intervalos correspondentes à função Booleana* $f = \overline{x}_2 + x_1\overline{x}_3$ *são* $[\emptyset, W - \{x_2\}]$ *e* $[\{x_1\}, W - \{x_3\}]$ *(ou simplificando a notação,* $[000, 101]$ *e* $[100, 110]$*)*. $\diamond$

Esta regra permite que as funções Booleanas sejam facilmente convertidas para a expressão correspondente no espaço Ψ_W.

3.4.2 Janela Mínima

Dizer que um operador é caracterizado localmente por uma janela W é dizer que o valor da transformação de um ponto por este operador depende apenas dos valores de um conjunto de pontos da sua vizinhança, delimitados pela janela W.

Proposição 3.4.1 (Janela Mínima) *Sejam $W, W' \subseteq E$, $W \subseteq W'$. Se $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ é um operador i.t., localmente definido por W, então $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ é também localmente definido por W'.* $\diamond$

Pela proposição acima, se ψ é localmente definido por uma janela W, então é também localmente definido por qualquer outra janela W' tal que $W \subseteq W'$. Isto garante a existência de uma janela mínima, a menor janela que é suficiente para caracterizar a operação desejada. A vantagem de utilizar a janela mínima está no fato de que os elementos do núcleo são elementos de $\mathcal{P}(W)$ e, portanto, quanto menor a janela W, menor é o conjunto $\mathcal{P}(W)$. No melhor caso, esta janela pode ser unitária (por exemplo, no caso do cálculo do complemento de uma imagem binária) e, no pior caso, pode ser uma janela do tamanho do domínio E.

3.5 Decomposição de Operadores com Propriedades Especiais

A caracterização de propriedades sobre os operadores define classes de operadores. Algumas destas classes possuem características algébricas de decomposição mais simples que a decomposição canônica. Apresentaremos as decomposições de duas classes particulares : operadores crescentes e operadores anti-extensivos.

3.5.1 Operadores Crescentes

A classe de operadores **crescentes** consiste de operadores que preservam a relação de inclusão, isto é, ψ é um operador crescente se $X \subseteq Y \Longrightarrow \psi(X) \subseteq \psi(Y), \quad \forall X, Y \in \mathcal{P}(E)$. Estes operadores podem ser decompostos como uniões de erosões e, do ponto de vista prático, caracterizam transformações que "preservam o tamanho relativo entre as imagens".

Teorema 3.5.1 (Representação de Matheron [Mat75]) *Seja $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ um operador i.t., localmente definido por uma janela W e crescente. Então, $\forall X \in \mathcal{P}(E)$,*

$$\psi(X) = \bigcup \{\varepsilon_A(X) : A \in \mathcal{K}_W(\psi)\}$$

A expressão acima pode ser reescrita da seguinte forma :

$$\psi(X) = \bigcup\{\varepsilon_A(X) : A \in Bas[\psi]\}$$

onde $Bas[\psi]$ é a classe minimal (não redundante) dos elementos de $\mathcal{K}_W(\psi)$, isto é, para qualquer $B \in \mathcal{K}_W(\psi)$, existe $B' \in Bas[\psi]$, $B' \subseteq B$ e não existe, em $Bas[\psi]$, um elemento contido propriamente em outro.

Observe que a decomposição acima é um caso particular da decomposição canônica, isto é,

$$A \in Bas[\psi] \Longleftrightarrow [A, W] \in \mathcal{B}_W(\psi).$$

3.5.2 Operadores Anti-extensivos

Um operador ψ é um operador **anti-extensivo** se $\psi(X) \subseteq X$, $\forall X \in \mathcal{P}(E)$. Neste caso, todos os elementos estruturantes que compõe o seu núcleo contém a origem, e do ponto de vista prático, estes operadores caracterizam transformações cujo "resultado é um subconjunto da imagem anterior à transformação".

O conjunto de pontos que pertencem a X e não pertencem a $\psi(X)$ pode ser expresso pelo conjunto $Z = X - \psi(X)$. O conjunto Z pode ser entendido como os pontos que são "eliminados" pelo operador.

Baseado neste fato, todo operador anti-extensivo, i.t. e localmente definido por uma janela W, possui uma representação definida da seguinte forma :

$$\psi(X) = X - \bigcup_{[A,B] \in \mathcal{K}^-} \lambda^W_{(A,B)}(X)$$

onde $\mathcal{K}^-$ é a classe de elementos estruturantes que define um operador que marca os pontos de Z. Em outras palavras, para cada par $[A, B] \in \mathcal{K}^-$, $o \in B$, $[A, B]$ localiza pontos de Z.

No próximo capítulo apresentaremos idéias relacionados ao projeto de operadores morfológicos ótimos, baseados nos resultados de decomposição. Resultados referentes à MM para reticulados completos quaisquer, não necessariamente o reticulado das partes de um conjunto, podem ser encontrados em [Ser88, BB93].

Capítulo 4

Projeto de Operadores

A característica de decomposição dos operadores morfológicos em termos de operadores elementares introduziu mais uma área de pesquisa em *MM*, relacionada ao projeto de operadores.

Um dos primeiros estudos nesta área é o trabalho de E.R.Dougherty ([Dou92a, Dou92b]). Dougherty estuda o projeto de Filtros Morfológicos ótimos, inserindo-os dentro do contexto de estimação estatística.

O objetivo deste capítulo é apresentar um resumo do trabalho realizado por Dougherty nesta linha de pesquisa, em relação às imagens binárias. Contudo, a colocação dos operadores no contexto estatístico necessita ser justificada e, portanto, dedicamos uma parte inicial deste capítulo para a apresentação de alguns conceitos e resultados relativos a "Random Sets" (conjuntos aleatórios).

4.1 Conjuntos Aleatórios

De uma forma geral, as técnicas utilizadas em Processamento de Imagens podem ser classificadas como estruturais ou estocásticas : as primeiras exploram as características geométricas das figuras nas imagens, enquanto as últimas exploram a natureza aleatória das mesmas. A maior parte delas explora apenas uma destas características. No caso de técnicas estocásticas podemos citar aqueles que se baseiam em campos Markovianos, variáveis aleatórias vetoriais, etc. Por outro lado, a *MM* é uma técnica que explora amplamente as características geométricas.

E quanto às técnicas que exploram simultaneamente as duas características ? Elas existem ? O estudo de Conjuntos Aleatórios ("random sets"), uma teoria introduzida independentemente por Matheron ([Mat75]) e Kendall ([Ken74]) na década de 70, e sua relação com a *MM* permite esta conjugação de características. A formulação apresentada por Matheron é mais interessante dentro do contexto de processamento de imagens, pois encontra-se diretamente associada à *MM*.

Nesta seção apresentaremos conceitos e resultados sobre Conjuntos Aleatórios Discretos e sua relação com a *MM*, baseados em um artigo de Goutsias ([Gou92]).

4.1.1 Conceitos Básicos

Considere um domínio $E \subseteq \mathbb{Z}^2$. Cada ponto $(i,j) \in E$ pode ser modelado por uma variável aleatória binária $\boldsymbol{x}_{i,j}$ que toma valores em $x_{i,j} \in \{0,1\}$, segundo uma distribuição de probabilidade P. A $|E|$-upla $\boldsymbol{x} = (\boldsymbol{x}_{i,j})_{(i,j)\in E}$, juntamente com a distribuição de probabilidade $P[\boldsymbol{x} = x]$ para cada $x \in \{0,1\}^{|E|}$, define uma variável aleatória vetorial (BRF). Muitas das técnicas que realizam análises estatísticas sobre imagens utilizam BRF's.

Um **conjunto aleatório discreto (DRS)** $\boldsymbol{X}$ em E é definido por

$$\boldsymbol{X} = \{(i,j) \in E : \boldsymbol{x}_{i,j} = 1\}. \tag{4.1}$$

Observe que, dado $x \in \{0,1\}^{|E|}$,

$$X = \{(i,j) \in E : x_{i,j} = 1\}$$

O DRS $\boldsymbol{X}$ em E é estatisticamente equivalente ao BRF $\boldsymbol{x}$ em E, isto é, $P[\boldsymbol{x} = x] = P[\boldsymbol{X} = X]$, para cada $x \in \{0,1\}^{|E|}$.

Seja $S \subseteq E$, finito. A função de distribuição acumulada de um DRS $\boldsymbol{X}$ é definida por

$$F_{\boldsymbol{X}}(B) = 1 - P[\boldsymbol{X} \supseteq B], \ \forall B \in \mathcal{P}(S) \tag{4.2}$$

De fato, pode-se verificar que

$$0 \leq F_{\boldsymbol{X}}(B) \leq 1, \quad \forall B \in \mathcal{P}(S),$$
$$F_{\boldsymbol{X}}(\emptyset) = 0,$$
$$\text{e } F_{\boldsymbol{X}}(B_1) \leq F_{\boldsymbol{X}}(B_2), \quad \forall B_1, B_2 \in \mathcal{P}(S), B_1 \subseteq B_2.$$

Das equações 4.1 e 4.2 temos que, para cada $B \in \mathcal{P}(S)$,

$$F_{\boldsymbol{X}}(B) = 1 - \sum_{x:X \supseteq B} P[\boldsymbol{x} = x]. \tag{4.3}$$

Vamos supor que $P[\boldsymbol{X} \in \mathcal{P}(S)] = 1$. A partir das equações 4.2 e 4.3, verifica-se que para cada BRF $\boldsymbol{x}$ e seu correspondente DRS $\boldsymbol{X}$ vale

$$P[\boldsymbol{x} = x] = P[\boldsymbol{X} = X] = \sum_{X' \subseteq X^c} (-1)^{|X'|}[1 - F_{\boldsymbol{X}}(X \cup X')], \ \forall X \in \mathcal{P}(S) \tag{4.4}$$

onde X^c é o complemento de X em $\mathcal{P}(S)$. $\diamond$

As equações 4.3 e 4.4 estabelecem uma relação biunívoca entre a distribuição de probabilidade do BRF $\boldsymbol{x}$ e a distribuição de probabilidade acumulada do DRS $\boldsymbol{X}$.

4.1.2 Estacionaridade

Um DRS $\boldsymbol{X}$ é **estacionário de ordem $\boldsymbol{n}$**, $n \geq 1$, se a sua função de distribuição acumulada $F_{\boldsymbol{X}}(B)$ é invariante sob translação de $\boldsymbol{X}$, para todo $B \in \mathcal{P}(S)$, tal que $|B| = n$, isto é,

$$F_{\boldsymbol{X}+z}(B) = F_{\boldsymbol{X}}(B), \quad \forall B \in \mathcal{P}(S), |B| = n \text{ e } \forall z \in E.$$

Uma vez que $B \subseteq \boldsymbol{X} + z \iff B - z \subseteq \boldsymbol{X}$, a condição acima é equivalente a : $F_{\boldsymbol{X}}(B + z) = F_{\boldsymbol{X}}(B)$, $\forall B \in \mathcal{P}(S), |B| = n$ e $\forall z \in E$.

Um DRS é **estritamente estacionário** se ele é estacionário de ordem n para cada n, $n \geq 1$.

Seja $W \subseteq S$, finito. Seja P a distribuição de probabilidade de $\boldsymbol{X}$ e P_{W_z} a distribuição de probabilidade de $\boldsymbol{X}_{W_z} = \boldsymbol{X} \cap W + z$ (em particular, P_W é a distribuição de probabilidade de $\boldsymbol{X}_{W_z}$ na origem).

Teorema 4.1.1 *Seja $\boldsymbol{X}$ um DRS estritamente estacionário. Então,*

$$F_{\boldsymbol{X}}(B) = 1 - P_W(\boldsymbol{X}_W \supseteq B), \quad \forall B \in \mathcal{P}(W).$$

onde $\boldsymbol{X}_W = \boldsymbol{X} \cap W$.

Dem.: $\forall B \in \mathcal{P}(W)$ e $z \in E$,

$$\begin{aligned}
F_{\boldsymbol{X}}(B) &= F_{\boldsymbol{X}}(B + z) \\
&= 1 - P(\boldsymbol{X} \supseteq B + z) \\
&= 1 - P(\boldsymbol{X} - z \supseteq B) \\
&= 1 - P_{W_z}(\boldsymbol{X} - z \cap W \supseteq B) && \text{(pois } B \subseteq W) \\
&= 1 - P_W(\boldsymbol{X} \cap W \supseteq B) && (\boldsymbol{X} \text{ é estritamente estacionário}) \\
&= 1 - P_W(\boldsymbol{X}_W \supseteq B)
\end{aligned}$$

$\diamond$

Este teorema mostra que para qualquer $B \in \mathcal{P}(W)$, $F_{\boldsymbol{X}}(B)$ pode ser caracterizada pela distribuição de probabilidade associada à variável aleatória $\boldsymbol{X} \cap W = \boldsymbol{X}_W$. A vantagem disto é que o conjunto das possíveis realizações de $\boldsymbol{X}_W$ é muito menor do que o de $\boldsymbol{X}$.

4.1.3 Transformações Morfológicas de Conjuntos Aleatórios

Uma forma que permite estudar as conexões do DRS com a *MM*, é a investigação das relações entre $P[\boldsymbol{X} \cap B \neq \emptyset]$ e $F_{\boldsymbol{X}}(B)$. O funcional de capacidade $T_{\boldsymbol{X}}$ de um DRS $\boldsymbol{X}$ é definido por

$$T_{\boldsymbol{X}}(B) = P[\boldsymbol{X} \cap B \neq \emptyset], \ B \in \mathcal{P}(S). \tag{4.5}$$

Verifica-se que $T_{\boldsymbol{X}}(B) = F_{\boldsymbol{X}^c}(B)$ (isto é, $T_{\boldsymbol{X}}(B)$ é a função de distribuição acumulada de $\boldsymbol{X}^c$). Tanto $F_{\boldsymbol{X}}(B)$ como $T_{\boldsymbol{X}}(B)$ podem ser calculados a partir de $P[\boldsymbol{x} = x]$.

A modelagem de imagens binárias por conjuntos aleatórios torna possível o estudo das imagens no contexto estatístico e, portanto, permite a criação de operadores baseados em técnicas e critérios estatísticos. Segundo as observações apresentadas na seção anterior, o DRS é equivalente ao BRF. Por que há, então, necessidade de definirmos conjuntos aleatórios em termos de vetores aleatórios ? A grande vantagem de encararmos imagens como conjuntos aleatórios e não como vetores aleatórios está no fato de que os conjuntos preservam a forma geométrica.

Supondo que as transformações sobre as imagens serão realizadas por operadores da *MM*, de que maneira eles afetam o conjunto aleatório que modela as imagens ?

Seja $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ um operador i.t. Se $\boldsymbol{X}$ é um DRS, então $\boldsymbol{Y} = \psi(\boldsymbol{X})$ é também um DRS, uma vez que no caso discreto $\psi(\cdot)$ é sempre mensurável ([Ser82]). Para caracterizar o efeito de $\psi(\cdot)$ sobre $\boldsymbol{X}$, é desejável estudarmos $F_{\boldsymbol{Y}}(B)$ e $T_{\boldsymbol{Y}}(B)$. Baseado na equação 4.5, temos

$$T_{\boldsymbol{Y}}(B) = P[\psi(\boldsymbol{X}) \cap B \neq \emptyset], \forall B \in S. \tag{4.6}$$

O interesse neste ponto é investigar a relação existente entre $T_{\boldsymbol{Y}}(B)$ e $T_{\boldsymbol{X}}(B)$. Algumas destas relações estão sendo investigadas. Em [Gou92], podemos encontrar as seguintes relações.

$$\begin{aligned}
T_{\boldsymbol{X} \cap W}(B) &= T_{\boldsymbol{X}}(B \cap W), \forall W \in \mathcal{P}(E) \\
T_{\boldsymbol{X}^c}(B) &= F_{\boldsymbol{X}}(B) \\
F_{\boldsymbol{X} \cup W}(B) &= F_{\boldsymbol{X}}(B \cap W^c), \forall W \in \mathcal{P}(E) \\
T_{\boldsymbol{X} \oplus A}(B) &= T_{\boldsymbol{X}}(B \oplus A^t) \\
F_{\boldsymbol{X} \ominus A}(B) &= F_{\boldsymbol{X}}(B \oplus A)
\end{aligned}$$

Estas fórmulas ilustram alguns casos particulares nos quais é possível caracterizar estatisticamente o DRS $\boldsymbol{Y} = \psi(\boldsymbol{X})$ a partir das características estatísticas do DRS $\boldsymbol{X}$.

Resumindo, uma vez que existe uma relação biunívoca entre os BRF's e os DRS's, estabelecida pelas equações 4.3 e 4.4, todos os conceitos definidos sobre variáveis aleatórias (tais como distribuição de probabilidade, distribuição de probabilidade acumulada, momentos, correlação, etc) podem ser definidas também para os conjuntos aleatórios. Portanto, as imagens encaradas como conjuntos aleatórios podem ser inseridas no contexto estatístico.

A conexão entre DRS's e *MM* fica estabelecida através da investigação de relações do tipo $\boldsymbol{X} \cap B \neq \emptyset$, $\boldsymbol{X} \cap W$, $\boldsymbol{X}$ e $\psi(\boldsymbol{X})$, etc. Os resultados obtidos a partir destes estudos podem ser empregados para projetar operadores ψ com caracerísticas estruturais e estocásticas desejadas (veja por exemplo, [Gou92]).

4.2 Projeto de Filtros Morfológicos Ótimos

Nesta seção descreveremos as principais idéias e resultados sobre projetos de filtros morfológicos ótimos, introduzidos por E. R. Dougherty, que tiveram início com os artigos [Dou92a] e [Dou92b].

A noção de filtros morfológicos foi introduzida por Serra ([Ser82]) e é amplamente discutida, por exemplo, em [Ser88]. Um **filtro morfológico** é um operador crescente ($X \subseteq Y \rightarrow \psi(X) \subseteq \psi(Y)$, $\forall X, Y \in \mathcal{P}(E)$) e idempotente ($\psi(\psi(X)) = \psi(X)$, $\forall X \in \mathcal{P}(E)$).

Dougherty utiliza o termo filtro morfológico crescente para referir-se aos operadores i.t. e crescentes, mas não necessariamente idempotentes. Ao longo dos seus trabalhos, o termo filtro é utilizado, de uma forma geral, para referir-se aos operadores aplicados em problemas de restauração, e o termo filtro binário é utilizado para referir-se aos filtros aplicados sobre imagens binárias.

4.2.1 Definição do Problema

Consideremos um vetor aleatório binário n-dimensional $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$, uma função de $\boldsymbol{x}$, denotado por $\boldsymbol{y}$, e um estimador g de $\boldsymbol{y}$.

Seja $\hat{\boldsymbol{y}} = g(\boldsymbol{x})$ o estimador de $\boldsymbol{y}$. Então, o erro médio quadrático (MSE) de $\hat{\boldsymbol{y}}$ é definido por :

$$MSE\langle\hat{\boldsymbol{y}}\rangle = E[|\hat{\boldsymbol{y}} - \boldsymbol{y}|^2]. \tag{4.7}$$

Em relação ao MSE, o estimador ótimo é aquele que minimiza a esperança definida acima. Sabe-se que, entre todos os estimadores, aquele que minimiza o MSE é a esperança condicional, definida por :

$$\hat{\boldsymbol{y}} = E[\boldsymbol{y}/\boldsymbol{x}].$$

O erro médio absoluto (MAE) de $\hat{\boldsymbol{y}}$ é definido por :

$$MAE\langle \hat{\boldsymbol{y}} \rangle = E[|\hat{\boldsymbol{y}} - \boldsymbol{y}|]. \qquad (4.8)$$

Como as variáveis aleatórias consideradas são binárias, é imediato que MSE e MAE são equivalentes. Se desejarmos que $\hat{\boldsymbol{y}}$ tome valores em $\{0,1\}$, então a esperança acima deve ser arredondada para 1 caso ela seja maior ou igual a 0.5 e para 0 caso contrário.

Suponha uma imagem X e um processo de ruído aleatório sobre X que produz uma outra imagem X' (X' é a imagem X corrompida pelo ruído). Dougherty propõe métodos baseados em estimação estatística para encontrar um filtro ψ tal que $\psi(X')$ seja "semelhante" a X.

Para considerarmos os operadores morfológicos no contexto da estimação estatística, deveremos encarar as imagens binárias como objetos aleatórios. Em geral, os vetores aleatórios são utilizados para modelar as imagens.

Seja $\boldsymbol{x}$ o vetor aleatório que modela as imagens de um determinado domínio $E \subseteq \mathbb{Z}^2$, e suponha a existência de um processo de ruído sobre $\boldsymbol{x}$, que caracteriza um outro vetor aleatório $\boldsymbol{x}'$. Suponha também que $\boldsymbol{x}$ e $\boldsymbol{x}'$ são estritamente estacionários.

Consideremos uma janela de observação W, tal que $|W| = n$. Seja $\boldsymbol{x}'_{W_z}$ o vetor aleatório associado aos pontos observados pela janela W quando seu centro encontra-se sobre um ponto z em E arbitrário. Seja $\boldsymbol{X}'_{W_z}$ o conjunto aleatório (veja equação 4.1) correspondente ao vetor aleatório $\boldsymbol{x}'_{W_z}$.

Sob a hipótese de estacionaridade estrita, $\boldsymbol{x}'_{W_z}$ e $\boldsymbol{x}'_W$ (ou equivalentemente, $\boldsymbol{X}'_{W_z}$ e $\boldsymbol{X}'_W$) são equivalentes (Teorema 4.1.1). No restante deste capítulo, vamos supor que $\boldsymbol{x}'$ é estritamente estacionário.

O problema de filtros ótimos envolve um processo de escolha de um melhor estimador (isto é, aquele que minimiza o erro) dentro de uma classe de estimadores. Dougherty investiga os filtros crescentes (baseados no Teorema de Representação de Matheron) e os filtros não-crescentes (baseados no Teorema da Decomposição Canônica). O critério estatístico utilizado para medida de erro é o MAE. Dougherty modela as imagens por BRF's ($\boldsymbol{x}'$) e não por DRS's ($\boldsymbol{X}'$). Apresentamos os resultados usando a formulação baseada em DRS e também em BRF.

4.2.2 Filtros Crescentes

Lembremos que qualquer operador ψ, i.t. e crescente, pode ser expresso como união de erosões, isto é (veja seção 3.5) :

$$\psi(X') = \bigcup\{\varepsilon_B(X') : B \in Bas[\psi]\}. \qquad (4.9)$$

Seja $\boldsymbol{X}'_W$ o DRS correspondente ao BRF $\boldsymbol{x}'_W$. Uma realização X'_W de $\boldsymbol{X}'_W$ é um conjunto e, portanto

$$f_{\varepsilon_B}(\boldsymbol{X}'_W) = \begin{cases} 1 & \text{se } B \subseteq \boldsymbol{X}'_W \\ 0 & \text{caso contrário.} \end{cases}$$

Em termos do BRF $\boldsymbol{x}'_W$, podemos reescrever

$$f_{\varepsilon_B(\boldsymbol{X}'_W)} = \min_j\{(\boldsymbol{x}'_W)_j : b_j = 1\}$$

onde $B = (b_1, b_2, ..., b_n)$. Note que $f_{\varepsilon_B}(\boldsymbol{X}'_W) = 1 \iff (b_j = 1 \implies (\boldsymbol{x}'_W)_j = 1)$, isto é, $f_{\varepsilon_B}(\boldsymbol{X}'_W) = 1 \iff B \subseteq \boldsymbol{X}'_W$.

Adaptando esta formulação para 4.9, temos :

$$\psi(\boldsymbol{X}') = \max_i\{f_{\varepsilon_{B_i}}(\boldsymbol{X}')\} = \max_i\{\min_j\{\boldsymbol{x}'_j : (b_i)_j = 1\}\}, \tag{4.10}$$

onde $\{B_i\}$ é a base de ψ.

Portanto, quando consideramos um filtro crescente ψ como estimador, o MAE pode ser expresso por :

$$MAE\langle\psi\rangle = E[|Y - \psi(\boldsymbol{X}'_W)|] = E[|Y - \max_i\{f_{\varepsilon_{B_i}}(\boldsymbol{X}'_W)\}|].$$

O filtro crescente ótimo é aquele que minimiza o MAE definido acima. As bases dos filtros crescentes são formadas por elementos estruturantes contidos em W. Como existem 2^n elementos estruturantes em W, escolher a melhor base corresponde a escolher dentre todas as subcoleções destes 2^n elementos, aquele que minimiza o MAE.

Loce ([Loc93], seção 2.3) apresenta o Teorema do MAE que permite calcular o MAE associado a um operador a partir do MAE de cada um dos elementos estruturantes que caracterizam a sua base. A partir deste teorema é derivada também uma forma mais eficiente de cálculo, bem adaptada para processos computacionais, conforme segue :

$$MAE\langle\psi_n\rangle = MAE\langle\psi_{n-1}\rangle + MAE\langle B_n\rangle - MAE\langle\phi_{n-1}\rangle.$$

onde

$$\begin{aligned} Bas[\psi_{n-1}] &= \{B_1, B_2, ..., B_{n-1}\} \\ Bas[\psi_n] = Bas[\psi_{n-1}] \cup \{B_n\} &= \{B_1, B_2, ..., B_n\} \\ Bas[\phi_{n-1}] &= \{B_1 \cup B_n, B_2 \cup B_n, ..., B_{n-1} \cup B_n\} \end{aligned}$$

Exemplo 4.2.1 *No caso de um operador ψ com três elementos na base, $Bas[\psi] = \{B_1, B_2, B_3\}$, o cálculo pelo teorema e pela forma mais eficiente são, respectivamente,*

$$\begin{aligned} MAE\langle B_1, B_2, B_3\rangle \;=\; & MAE\langle B_1\rangle + MAE\langle B_2\rangle + MAE\langle B_3\rangle \\ & -MAE\langle B_1 \cup B_2\rangle - MAE\langle B_1 \cup B_3\rangle - MAE\langle B_2 \cup B_3\rangle \\ & +MAE\langle B_1 \cup B_2 \cup B_3\rangle \end{aligned}$$

e

$$MAE\langle\psi_n\rangle \quad = \quad MAE\langle B_1, B_2\rangle + MAE\langle B_3\rangle - MAE\langle B_1 \cup B_3, B_2 \cup B_3\rangle.$$

Este teorema permite que o MAE possa ser calculado recursivamente, para bases cada vez maiores, a partir do MAE associado a subconjuntos próprios da base para o qual se deseja calcular o MAE.

◇

Abordagem baseada em restrições

Devido a quantidade enorme de possíveis bases, testar todas as possibilidades e escolher a de menor MAE não é uma tarefa viável. Frente a este problema, algumas abordagens que procuram reduzir o espaço e tempo de busca são apresentadas em [Loc93].

Em relação às abordagens que restringem o espaço de busca, são considerados basicamente os seguintes tipos de tratamento :

- uma das possíveis restrições que podem ser impostas é a restrição algébrica. Por exemplo, pode-se requerer que o operador a ser estimado seja crescente e anti-extensivo. No caso de operadores anti-extensivos, cada elemento da base deve conter a origem, e portanto, a busca de todas as possibilidades pode ser restrita sobre todas as bases que integram apenas elementos estruturantes que contém a origem.

- Um outra restrição diz respeito ao tamanho da base. Em vez de testar todas as possíveis bases, testam-se todas as possíveis bases que integram no máximo m elementos. (veja [DL94]).

- Outra restrição refere-se ao tamanho da janela de observação. Uma vez que o número de elementos estruturantes possíveis é $2^{|W|}$, utilizar $W' \subseteq W$ em vez de W, reduz este espaço de busca. (veja [Loc93], página 33).

- pode-se considerar também uma biblioteca de elementos estruturantes. Isto é, em vez de realizar a busca sobre o conjunto de todos os possíveis elementos estruturantes, consideram-se subgrupos deste, organizados como bibliotecas especializados para problemas específicos (veja [Loc93], página 35).

4.2.3 Filtros não-crescentes

Dougherty apresenta uma outra abordagem quando trata de filtros i.t. não necessariamente crescentes. Este tratamento é baseado na decomposição canônica, apresentada no capítulo anterior.

De forma análoga ao caso da erosão, pode-se aplicar um operador sup-gerador sobre o conjunto aleatório $\boldsymbol{X}'_W$.

$$(f_{\lambda^W_{(A,B)}}(\boldsymbol{X}'_W) = 1) \Longleftrightarrow (A \subseteq \boldsymbol{X}'_W \subseteq B) \Longleftrightarrow (A \subseteq \boldsymbol{X}'_W \text{ e } B^c \subseteq (\boldsymbol{X}'_W)^c).$$

Em termos do BRF $\boldsymbol{x}'_W$ podemos reescrever

$$f_{\lambda^W_{(A,B)}}(\boldsymbol{X}'_W) = \min\{\min_j\{(\boldsymbol{x}'_W)_j : A_j = 1\}, \min_j\{(\boldsymbol{x}'_W)^c_j : b^c_j = 1\}\}.$$

Portanto um filtro ψ i.t., adaptado à sua forma canônica, pode ser expresso por[1] :

$$f_\psi(\boldsymbol{X}') = \max_i\{\min\{\min_j\{\boldsymbol{x}'_j : (A_i)_j = 1\}, \min_j\{(\boldsymbol{x}')^c_j : (b^c_i)_j = 1\}\}\}. \tag{4.11}$$

Existem 2^n possíveis realizações X'_W de $\boldsymbol{X}'_W$, para as quais $f_\psi(X'_W) = 1$ ou $f_\psi(X'_W) = 0$. Observe que cada realização X'_W de $\boldsymbol{X}'_W$ pode ser representada pelo par $[X'_W, X'_W]$. Uma coleção de pares $[X'_W, X'_W]$ caracteriza um estimador dado pela expressão 4.11. Assim, a escolha de um filtro i.t. ótimo corresponde à escolha de uma coleção de pares $[X'_W, X'_W]$ que minimiza o MAE, dentre todos as possíveis coleções.

Para selecionar os pares de elementos estruturantes $[A, B]$, consideram-se realizações X' de $\boldsymbol{X}'$. Ao se transladar W sobre X', obtemos as realizações de $\boldsymbol{X}'_W$. Para cada realização X'_W de $\boldsymbol{X}'_W$ armazena-se o valor de X no ponto z, X_z, juntamente com o par $[X'_W, X'_W]$. O par $[X'_W, X'_W]$ é integrado na decomposição da equação 4.11 se o número de 1's armazenados excede o de 0's (este procedimento corresponde justamente ao arredondamento de $E[\boldsymbol{y}/\boldsymbol{x}]$, para tornar o estimador uma função binária).

Os pares $[X'_W, X'_W]$ selecionados correspondem aos mintermos da função Booleana isomorfa ao operador estimado. Aplicando-se operações lógicas (por exemplo, o algoritmo de Quine-McCluskey) sobre estes mintermos, pode-se reduzir o número de termos na expressão 4.11. (Em outras palavras, os pares selecionados correspondem exatamente aos elementos de $\mathcal{K}_W(\psi)$ e as operações de redução lógica, no seu limite, resultam na base do operador).

No caso dos filtros crescentes, o cálculo do MAE depende da distribuição de probabilidade conjunta de $(\boldsymbol{X}', \boldsymbol{X})$. Ela é estimada a partir de realizações (X', X) de $(\boldsymbol{X}', \boldsymbol{X})$. No caso dos filtros não-crescentes, os pares $[X'_W, X'_W]$ são também selecionados de acordo com esta distribuição estimada. Em [DL94], podemos encontrar algumas discussões que comparam o desempenho de filtros crescentes e não-crescentes obtidos a partir de uma distribuição estimada.

[1] Neste ponto há uma pequena diferença de notação entra a empregada por Dougherty e a aqui empregada : Dougherty utiliza o conceito de operadores "hit or miss", que são equivalentes às sup-geradoras (com excessão de sua caracterização que é totalmente baseada em erosões)

Observe também que em ambos os casos, as expressões 4.10 e 4.11 referem-se ao filtro ótimo calculado sobre um ponto $z \in E$. Sob a hipótese de estacionaridade estrita, este filtro é o filtro ótimo sobre E, já que z é um ponto arbitrário de E.

Todos os problemas tratados por Dougherty são problemas de restauração (isto justifica por que ele utiliza o termo "filtro", em vez de operador, em todos os seus trabalhos), nos quais o critério estatístico para escolha do estimador ótimo é o MAE. Além disso, nota-se em seus trabalhos uma nítida distinção entre filtros crescentes e não-crescentes, para os quais ele realiza tratamento diferenciado.

Capítulo 5

Modelos de Aprendizado Computacional

Os esforços que buscavam aproveitar o poder de processamento dos computadores digitais, cada vez mais velozes, não excluiu o desejo de se criar uma "máquina inteligente", capaz de tomar decisões. Uma das áreas de pesquisa resultante destes esforços é uma teoria ainda recente conhecida por "Machine Learning" (Teoria de Aprendizado Computacional), que estuda métodos para modelar matematicamente os processos de aprendizado.

Alguns resultados significativos nesta área (teoria da decisão estatística [DH73]) foram obtidos na década de 70. Um fundamento teórico mais sólido foi estabelecido com a introdução do modelo PAC de aprendizado (do inglês "Probably Approximately Correct"), proposto por Valiant em [Val84].

Na primeira parte deste capítulo situaremos o processo de aprendizado dentro de uma modelagem matemática e na segunda parte apresentaremos os resultados referentes ao modelo PAC de Valiant, baseado no texto [AB92]. Na última seção deste capítulo, apresentaremos alguns resultados referentes à extensão do modelo PAC, tomando como referências principais os artigos de Haussler ([Hau92]) e Kearns e Schapiro ([KS90]).

5.1 Modelagem do Processo de Aprendizado

Informalmente, qualquer processo que é capaz de "aprender um conceito", a partir de exemplos que o ilustram, é denominado "aprendizado". Por exemplo, considere uma caixa com bolas de cor vermelha e de várias outras cores. Suponha que um aluno quer aprender o conceito "cor vermelha". Então serão sorteadas, ao acaso, algumas bolas desta caixa e o aluno receberá as informações "bola é vermelha" ou "bola não é vermelha" para cada bola sorteada. Baseado neste processo de treinamento, espera-se que o aluno torne-se capaz de distinguir a cor vermelha das demais cores.

Nesta seção, apresentaremos a modelagem do processo de aprendizado na qual está baseado o modelo PAC básico de Valiant, que consiste basicamente do aprendizado de "conceitos" que podem ser modelados por funções booleanas.

Consideremos um domínio D cujos elementos são as representações codificadas de um "mundo real". Por exemplo, as imagens binárias definidas sobre o conjunto $E \subseteq \mathbb{Z}^2$, onde $|E| = n$, podem ser representadas por vetores ou matrizes de bits de tamanho n. As letras do alfabeto podem ser representadas por um vetor de atributos que indicam por exemplo características topológicas, sua altura, largura, existência ou não de "perninhas" (como as letras g, j, q, p), existência de pingos (como a letra i, j), etc.

Seja $\sum$, denominado *alfabeto*, um conjunto para descrever elementos de D. Denotamos por $\sum^n$ o conjunto de todas as n-uplas de elementos de $\sum$.

Seja $X \subseteq \sum^n$. Um conceito c é uma função definida por $c : X \to \{0, 1\}$. X é denominado o *espaço de exemplos* e seus elementos são denominados *exemplos*. Um exemplo $x \in X$ é um *exemplo positivo* se $c(x) = 1$ e é um *exemplo negativo* se $c(x) = 0$. Dado o conjunto de todos os exemplos positivos, ele determina e é determinado por um conceito c. Por esta razão, um conceito c pode ser também encarado como um subconjunto de X.

O conjunto C de todos os possíveis conceitos é denominado *espaço de conceitos*. O objetivo de um processo de aprendizado é produzir um conceito h que seja uma boa aproximação de um conceito alvo $t \in C$, que queremos aprender. O conjunto de todos os conceitos que podem ser aprendidos é denominado *espaço de hipóteses* e denotado por H.

Uma *amostra de tamanho* m é uma sequência de m exemplos, ou seja, uma m-upla $\boldsymbol{x} = (x_1, x_2, ..., x_m) \in X^m$. Uma *amostra de treinamento* s *de tamanho* m é um elemento de $(X \times \{0, 1\})^m$, ou seja

$$s = ((x_1, b_1), (x_2, b_2), ..., (x_m, b_m))$$

onde $b_i = t(x_i)$, indica se os exemplos são positivos ou negativos relativamente ao conceito t. Dizemos que s é *consistente* se $x_i = x_j \Longrightarrow b_i = b_j, 1 \leq i, j \leq m$.

Um *algoritmo de aprendizado* é uma função L que associa a cada amostra de treinamento s consistente de tamanho m, relativo a um conceito alvo $t \in H$, uma hipótese $h \in H$. Denotamos $L(s) = h$.

Uma hipótese $h \in H$ é *consistente com uma amostra* s se $h(x_i) = b_i$ para cada $1 \leq i \leq m$. Um algoritmo de aprendizado é *consistente* se ele é consistente com todas as possíveis amostras de treinamento s consistentes.

5.2 Modelo de Aprendizado PAC

Dentro da modelagem apresentada na seção anterior, o modelo de aprendizado PAC, proposto por Valiant, considera $\sum = \{0, 1\}$. Além disso, supõe que os elementos do espaço de exemplos X não são contraditórios, ou seja, todas as amostras são consistentes. Vamos supor também que $C = H$.

Suponha que X define um espaço de probabilidades e seja μ a distribuição de probabilidades associada a X. Dado um conceito-alvo $t \in H$, definimos o erro de uma hipótese $h \in H$ relativamente a t, da seguinte forma :

$$er_{\mu}(h, t) = \mu\{x \in X : h(x) \neq t(x)\}.$$

O espaço X^m de amostras de treinamento de tamanho m herda a estrutura de espaço de probabilidades de X. Denotamos a distribuição de probabilidades correspondente a X^m por μ^m. Dado $Y \subset X^m$, interpretamos o valor $\mu^m(Y)$ como sendo 'a probabilidade de uma amostra aleatória de m exemplos, obtidos de X, segundo a distribuição μ, pertencer a Y'.

Seja $S(m, t)$ o conjunto de amostras de treinamento de tamanho m. Existe uma bijeção entre X^m e $S(m, t)$ expressa por uma função $\phi : X^m \to S(m, t)$, que a cada m-upla de exemplos $\boldsymbol{x} = (x_1, x_2, ..., x_m)$ associa a amostra $s = ((x_1, b_1), (x_2, b_2), ..., (x_m, b_m))$.

Logo, $\mu^m\{s \in S(m, t)\ :\ s$ tem a propriedade $P\}$ pode ser interpretada como $\mu^m\{\boldsymbol{x} \in X^m\ :\ \phi(x)$ tem a propriedade $P\}$. Isto permite calcularmos o erro da hipótese $h = L(s)$ e a probabilidade deste erro ser menor que um dado valor ϵ em função da distribuição de probabilidade sobre X.

Definição 5.2.1 *Um algoritmo L é PAC (Probably Approximately Correct) para um espaço de hipóteses H se, dados*

- *um número real δ $(0 < \delta < 1)$ e*
- *um número real ϵ $(0 < \epsilon < 1)$,*

então existe um inteiro positivo $m_0 = m_0(\delta, \epsilon)$ tal que

- *para qualquer conceito alvo $t \in H$, e*
- *para qualquer distribuição de probabilidade μ sobre X,*

sempre que $m \geq m_0$, $\mu^m\{s \in S(m, t) : er_{\mu}(L(s), t) < \epsilon\} > 1 - \delta$.

Estamos interessados em encontrar um algoritmo de aprendizado PAC, isto é, um algoritmo que produz uma hipótese h, tal que, com alta probabilidade $(1-\delta)$, o erro entre a hipótese h e o conceito-alvo t seja menor que ϵ. O termo "provalvelmente aproximadamente correto" está associado à esta idéia.

Dado $\epsilon \in (0, 1)$, o conjunto

$$B_\epsilon = \{h \in H \ : \ er_\mu(h, t) \geq \epsilon\}$$

é denominado o conjunto das hipóteses ϵ-ruins .

Definição 5.2.2 *Seja H um espaço de hipóteses e $H[s]$ o conjunto de todas as hipóteses de H consistentes com s. O espaço H é potencial se, dados ϵ e δ tais que $0 < \epsilon, \delta < 1$, existe um inteiro positivo $m_0 = m_0(\epsilon, \delta)$ tal que para qualquer $m \geq m_0$,*

$$\mu^m\{s \in S(m, t) \ : \ H[s] \cap B_\epsilon = \emptyset\} > 1 - \delta,$$

para qualquer distribuição de probabilidade μ sobre X e qualquer $t \in H$.

Teorema 5.2.1 *Qualquer espaço finito de hipóteses é potencial.* $\diamond$

Teorema 5.2.2 *Se H é um espaço de hipóteses potencial, e L é um algoritmo de aprendizado consistente, então L é PAC.* $\diamond$

As demostrações destes dois teoremas podem ser encontradas em [AB92]. A partir destes teoremas, é conseqüência imediata que qualquer algoritmo consistente quando considerado um espaço finito de hipóteses é PAC. Apresentamos a seguir, como ilustração, um exemplo de algoritmo PAC introduzido por G. Valiant ([Val84]).

Exemplo 5.2.1 *Seja $D_{n,k}$ o espaço das funções Booleanas de n variáveis e que podem ser expressas como a disjunção de monômios de tamanho máximo k ($1 < k < n$). Vamos supor que é dada uma amostra de treinamento de tamanho m.*

> *faça h = disjunção de todos os monômios de tamanho máximo k ;*
> *para $i = 1$ até m faça*
>   *se $b_i = 0$ e $h(x_i) = 1$*
>   *então elimine os monômios $\mathcal{M}$ de h tais que $\mathcal{M}(x_i) = 1$;*
> *$L(s) = h$;*

Observamos que, no algoritmo, inicialmente $h(x_i) = 1$ para todo x_i. A cada passo do algoritmo, se $b_i = 0$ e $h(x_i) = 1$ então os monômios $\mathcal{M}$ para os quais $\mathcal{M}(x_i) = 1$

são eliminados de h. Ao final do algoritmo $h(x_i) = b_i$ para todo $1 \leq i \leq m$. Logo este algoritmo é consistente e portanto é PAC.

$\diamond$

Em resumo, dado um espaço de exemplos sem contradições relativamente a um conceito-alvo t (i.e., no qual todas as amostras são consistentes) e um espaço finito de hipóteses, qualquer algoritmo de aprendizado L consistente é PAC. Isto é, existe um inteiro positivo m_0 tal que se tomarmos uma amostra de treinamento s, com mais de m_0 exemplos, a probabilidade do erro entre a hipótese $h = L(s)$ e o conceito-alvo t ser menor que ϵ é maior que $1 - \delta$.

Uma questão crucial nesta modelagem está associada ao número de exemplos m_0 necessários para garantir esta precisão ϵ e confiabilidade δ. Segundo [AB92], um limitante superestimado é dado por :

$$m_0(\delta, \epsilon) = \frac{1}{\epsilon} ln \frac{|H|}{\delta}$$

onde $|H|$ representa a cardinalidade do espaço de hipóteses.

5.3 Extensão do Modelo PAC

O modelo PAC de Valiant impõe restrições sobre o espaço de exemplos e sobre a classe de conceitos que podem ser aprendidos. Uma destas restrições é a suposição de que não existem exemplos contraditórios no espaço de exemplos X, isto é, dados dois exemplos $x_i, x_j \in X$, se $x_i = x_j$ então $b_i = b_j$.

No entanto, na prática, é comum encontrarmos situações nas quais os exemplos são contraditórios. Neste caso, pode-se supor que os exemplos são pares ordenados (x, b) onde x é gerado segundo uma distribuição de probabilidade μ sobre X conforme a descrição da seção anterior, porém o elemento b não é determinístico, isto é, é gerado segundo uma distribuição condicional[1] $p(b/x)$. A distribuição μ sobre X juntamente com a distribuição condicional $p(b/x)$ determinam naturalmente uma distribuição de probabilidade conjunta P sobre $X \times \{0, 1\}$.

Existem também outras restrições impostas pelo modelo PAC de Valiant (veja maiores detalhes em [Hau92, KS90]). No entanto, neste texto, abordaremos as extensões que estão relacionadas com a restrição citada acima.

Na situação considerada acima, estamos interessados em aprender uma função Booleana (decisão) h, que seja uma boa aproximação do espaço $X \times \{0, 1\}$, a partir de exemplos (x, b) gerados aleatoriamente segundo a distribuição de probabilidade conjunta P sobre $X \times \{0, 1\}$.

[1] Ver por exemplo [Mey69], seção 6.2.

Seja $h : X \rightarrow \{0,1\}$ uma hipótese (aqui será denominada decisão). No modelo PAC básico, o erro de h em relação a um conceito-alvo $t : X \rightarrow \{0,1\}$ é caracterizado em termos da probabilidade associada aos elementos $x \in X$ para os quais $h(x) \neq t(x)$. Haussler propõe uma generalização no qual o "erro" é caracterizado através de uma função l_h, denominada **função de perda**. A função de perda "mede" a perda cometida ao se escolher uma determinada decisão h, isto é, $l_h : X \times \{0,1\} \rightarrow [0,1]$ associa uma perda não negativa $l_h(x,b)$ para cada par (x,b) em $X \times \{0,1\}$.

O **risco** de uma hipótese $h \in H$, relativo à distribuição de probabilidade conjunta P sobre $X \times \{0,1\}$ e à função de perda l_h, é definido por

$$r_P(l_h) = E[l_h(x,b)].$$

Seja $h^* \in H$ uma hipótese de menor risco para o espaço de exemplos $X \times \{0,1\}$, segundo a função de perda l_h. Isto é,

$$r_P(l_{h^*}) \leq r_P(l_h), \ \forall h \in H.$$

Uma amostra de treinamento s de tamanho m ($s = \{(x_1,b_1),(x_2,b_2),...,(x_m,b_m)\}$) é definida da mesma forma, porém nesta generalização, cada exemplo (x_i,b_i) é obtido segundo a distribuição de probabilidade conjunta P sobre $X \times \{0,1\}$ (observe que neste caso as amostras não são necessariamente consistentes).

Seja $\boldsymbol{s}$ o processo aleatório que modela as amostras de treinamento de tamanho m, e seja P_m a distribuição de probabilidade associada a $\boldsymbol{s}$, em função de P.

Com base nestes novos conceitos, podemos redefinir a noção de algoritmo PAC num contexto mais genérico :

Definição 5.3.1 *Dizemos que um algoritmo L é PAC para um espaço de decisões H se, dados*

- *um número real δ ($0 < \delta < 1$) e*
- *um número real ϵ ($0 < \epsilon < 1$),*

então existe um inteiro positivo $m_0 = m_0(\delta,\epsilon)$ tal que

- *para qualquer espaço de exemplos $X \times \{0,1\}$,*
- *para qualquer distribuição de probabilidade conjunta P sobre $X \times \{0,1\}$*

sempre que $m \geq m_0$

$$P_m(|r_P(l_{L(\boldsymbol{s})}) - r_P(l_{h^*})| < \epsilon) > 1 - \delta.$$

◇

O objetivo de um processo de aprendizado é escolher uma decisão de menor risco entre todas as decisões em H. Para isto, deve-se escolher um critério de decisão que minimiza a função de perda l_h.

Vamos considerar a inexistência de exemplos contraditórios em $X \times \{0,1\}$ e a função de perda $l_h(x,b) = |h(x) - b|$. Então é fácil verificar que $r_P(l_{h^*}) = 0$ e

$$\begin{aligned} r_P(l_h) &= E[l_h(x,b)] \\ &= \sum_{(x,b)\in X\times\{0,1\}} |h(x) - b| P(x,b) \\ &= \sum_{x\in X} |h(x) - t(x)| \mu(x) \\ &= \mu^m(\{x \in X : h(x) \neq t(x)\}) \end{aligned}$$

Sob estas condições, a definição apresentada aqui reduz-se à definição de Valiant, apresentada na seção anterior.

Notemos que, para $l_h(x,b) = |h(x) - b|$, este modelo trata também o problema de restauração estudado por Dougherty (capítulo 4).

O estudo para determinar o valor m_0 da amostra de treinamento, neste caso mais geral, envolve vários conceitos teóricos complexos. Além disso, observa-se que o limite teórico apresentado em [Hau92] está muito acima dos valores observados na prática (veja capítulos 9 e 10). Por estes motivos, omitimos a sua apresentação aqui. Algumas discussões relacionadas com o tamanho da amostra de treinamento, sob um aspecto prático, serão apresentadas no capítulo 10.

Capítulo 6

Algoritmos para o Aprendizado de Funções Booleanas

No capítulo anterior, apresentamos algumas noções e resultados associados ao modelo PAC, que estuda o aprendizado de conceitos que podem ser modelados por funções Booleanas, e algumas generalizações baseadas naquele modelo.

Um algoritmo de aprendizado PAC trivial pode ser escrito da seguinte forma :

```
Entrada:  - Conjunto de exemplos S = {(x_i, b_i) : 1 ≤ i ≤ m}  .
Saída :   - uma hipótese (função Booleana) f

{
    f = ∅;            % conjunto vazio
    Para i de 1 até m faça
            se b_i = 1 então faça
                  f = f ∪ {x_i};
    Devolva f;
}
```

Este algoritmo fornece como resultado uma função Booleana na FND. É fácil observar que ele é PAC pois ele é consistente, isto é, para cada $(x_i, b_i) \in S$, se $b_i = 1$ então $f(x_i) = 1$ e se $b_i = 0$ então $f(x_i) = 0$.

Entretanto, na prática, representar uma função Booleana na FND não é interessante, uma vez que o número de termos e literais envolvidos geralmente é grande, comprometendo a sua manipulação. Em geral estamos interessados em encontrar uma expressão equivalente mais simples, que envolve um menor número de termos e literais.

Sabemos que uma expressão Booleana pode ser minimizada através de simplificação de expressões (ver capítulo 2). Em particular, as funções Booleanas expressas em FND podem ser minimizadas pelo algoritmo QM apresentado no capítulo 2. Entretanto, quando o número de variáveis Booleanas envolvidas é grande, este algoritmo torna-se inviável pois requer grande espaço de memória e tempo de processamento.

Motivados por esta razão, começamos a investigar algoritmos alternativos e desta investigação resultou um novo algoritmo que não encontramos na literatura consultada. Este capítulo será dedicado à apresentação deste novo algoritmo e também à avaliação de seu desempenho, inclusive em comparação ao desempenho do algoritmo QM.

Deste ponto em diante, os mintermos para os quais a função toma valor 1 serão denominados **mintermos positivos** e aqueles para os quais ela toma valor 0 serão denominados **mintermos negativos**, por uma convenção estabelecida neste texto para manter a associação entre mintermos e exemplos de treinamento considerados no capítulo 5. Em outras palavras, os exemplos positivos e negativos correspondem, respectivamente, aos mintermos positivos e negativos.

6.1 Um Novo Algoritmo para Minimização de Funções Booleanas

Seja f uma função Booleana com n variáveis expressa na FND. Estamos interessados em obter a sua representação cúbica minimal, isto é, sua representação em termos de um conjunto de sub-cubos maximais, em relação ao n-cubo definido no n-espaço. Observe que na representação cúbica minimal, os pontos pertencentes aos sub-cubos são exatamente os pontos correspondentes aos mintermos positivos de f.

No caso do algoritmo QM, esta representação é obtida pelo seguinte processo : inicialmente combinam-se todos os mintermos positivos dois a dois de modo a gerar os 1-cubos. A seguir estes 1-cubos são combinados dois a dois para gerar os 2-cubos e assim sucessivamente até que nenhuma combinação seja mais possível. Os sub-cubos que não puderam ser utilizados para gerar sub-cubos de dimensão maior constituem o conjunto de sub-cubos maximais no qual estamos interessados.

Neste novo algoritmo, o processo é inicializado com o n-cubo (que equivale à função Booleana constante 1) e a partir dele os pontos correspondentes aos mintermos negativos de f são eliminados sucessivamente. Após todos os mintermos negativos terem sido eliminados, os pontos restantes do n-cubo correspondem ao conjunto de mintermos positivos e mais aqueles que não foram definidos (não são positivos nem negativos).

Se encararmos o cubo como uma estrutura de reticulado Booleano, o n-cubo pode ser visto como um intervalo $[\emptyset, W]$, onde $|W| = n$. Neste caso, estamos interessados em obter a representação minimal de f em termos de um conjunto de intervalos maximais. O ponto chave deste novo algoritmo é a possibilidade de representar o conjunto de todos os pontos de um intervalo, a menos de um, em termos de um conjunto de intervalos maximais. Este resultado é formalizado pelo seguinte teorema.

Teorema 6.1.1 *Seja* $[A, B] \subseteq \mathcal{P}(W)$ *e* $P \in [A, B]$. *Então,*

$$[A, B] - \{P\} = \{[A, B \cap \{a\}^c] : a \in P\} \cup \{[A \cup \{b\}, B] : b \in P^c\},$$

onde $\cdot^c$ *é o complemento em relação a* W. *Além disso, os intervalos do lado direito da igualdade são maximais.*

Dem.:

$$\begin{aligned}
&Max([A, B] - \{P\}) \\
&= Max([A, B] - [P, P]) && (\{\mathrm{P}\} = [\mathrm{P,P}]) \\
&= Max([A, B] \cap [P, P]^c) \\
&= Max([A, B]) \sqcap Max([P, P]^c) && (\text{definição de } \sqcap) \\
&= \{[A, B]\} \sqcap (\{[\emptyset, \{a\}^c] : a \in P\} \cup \{[\{b\}, W] : b \in P^c\}) && (\text{Proposição 3.4 em [BS95]}) \\
&= Max([A, B] \cap (\{[\emptyset, \{a\}^c] : a \in P\} \cup \{[\{b\}, W] : b \in P^c\})) && (\text{definição de } \sqcap) \\
&= Max(\{[A, B \cap \{a\}^c] : a \in P\} \cup \{[\{b\} \cup A, B] : b \in P^c\}) && (\text{distributividade de } \cap) \\
&= \{[A, B \cap \{a\}^c] : a \in P\} \cup \{[\{b\} \cup A, B] : b \in P^c\}
\end{aligned}$$

◇

Corolário 6.1.1 *Seja* $[A, B] \subseteq \mathcal{P}(W)$ *e* $P \in [A, B]$. *Então,*

$$[A, B] - \{P\} = \{[A, B \cap \{a\}^c] : a \in P \cap A^c\} \cup \{[\{b\} \cup A, B] : b \in P^c \cap B\}.$$

Dem.: Decorre imediatamente do teorema anterior se observarmos que

se $a \in P$ e $a \in A \Longrightarrow [A, B \cap \{a\}^c] = \emptyset$ e se $b \in P^c$ e $b \in B^c \Longrightarrow [\{b\} \cup A, B] = \emptyset$.

◇

Corolário 6.1.2 *O número de intervalos na decomposição acima é exatamente igual à dimensão*[1] *do intervalo* $[A, B]$, *isto é,* $|max([A, B] - \{P\})| = dim([A, B])$.

[1] A definição de *dimensão de um intervalo* está na página 32

Dem.:

$$\begin{aligned} ||P \cap A^c|| + ||P^c \cap B|| &= (||P|| - ||A||) + (||B|| - ||P||) \quad \text{(pois } P \in [A, B]) \\ &= ||B|| - ||A|| \\ &= dim([A, B]) \end{aligned}$$

$\diamond$

6.1.1 Descrição do Algoritmo

Com base nos resultados acima, ao se extrair um mintermo negativo P_1 do intervalo inicial $[\emptyset, W]$, o conjunto de pontos correspondente a $[\emptyset, W] - \{P_1\}$ pode ser expresso por um conjunto $\mathcal{N}$ de intervalos maximais. Ao se extrair um segundo mintermo negativo P_2, este deve ser extraído de $[\emptyset, W] - \{P_1\}$ e não de $[\emptyset, W]$. Isto significa dizer que P_2 deve ser extraído de cada um dos intervalos de $\mathcal{N}$ que o contém. Este processo pode originar vários conjuntos de intervalos maximais, cuja união não necessariamente é um conjunto de intervalos maximais.

Para garantir que o conjunto de todos os intervalos (aqueles que foram gerados ao se extrair P_2 mais aqueles de $\mathcal{N}$ que não foram "particionados") é maximal, antes de extrair um terceiro exemplo negativo P_3, devem-se eliminar os intervalos que estão contidos em outros de dimensão maior. Neste caso, note que apenas os intervalos que foram gerados neste momento podem estar contidos em outros, que são necessariamente aqueles que não foram particionados neste momento.

Tendo-se este processamento adicional ao final da extração de cada mintermo negativo, obtém-se um conjunto de intervalos maximais, que correspondem a todos os elementos do intervalo inicial, menos aqueles que foram extraídos até o momento.

Portanto, ao final da extração de todos os mintermos negativos, obtém-se exatamente o conjunto de intervalos maximais correspondente à representação minimal de f (isto é, todos os implicantes primos de f).

Este processo de extração sucessivo de mintermos negativos está associado a uma idéia de particionamento sucessivo de intervalos, no qual, a cada passo, são gerados intervalos de dimensão menor. Baseado nesta observação, poderíamos sugerir um nome como, por exemplo, "Particionamento Sucessivo de Intervalos" para este novo algoritmo. Por motivos de simplificação e conveniência, decidimos adotar o nome "ISI" (do inglês "incremental splitting of intervals").

A seguir descrevemos os passos do algoritmo ISI.

- <u>Passo 0</u> : Seja $\mathcal{T}$ o conjunto de intervalos iniciais (usualmente $[\emptyset, W]$). Separe os mintermos negativos dos positivos. Se não existir nenhum mintermo negativo, então devolver $\mathcal{T}$. Se não existir nenhum mintermo positivo, então devolver $\emptyset$.

- Passo 1 : Separe os intervalos de $\mathcal{T}$ pelo seguinte critério : colocar os intervalos de $\mathcal{T}$ que contém o próximo mintermo negativo a ser extraído (P) em $\mathcal{T}_0$ e os que não contém em $\mathcal{T}_1$. E faça $\mathcal{N} = \emptyset$.

- Passo 2 : particionar cada um dos intervalos de $\mathcal{T}_0$ e gerar os subintervalos. Os subintervalos que não estiverem contidos em algum intervalo de $\mathcal{T}_1$ serão colocados em $\mathcal{N}$.

- Passo 3 : faça $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{N}$ e repita os passos 1 a 3 para o próximo mintermo negativo, enquanto existirem.

O algoritmo ISI

Descrevemos a seguir o algoritmo ISI em pseudo-código :

```
Entrada: - Conjunto de mintermos positivos e negativos
         - Intervalos de inicialização
Saída :  - Conjunto de implicantes primos essenciais

{
    T = intervalos de inicialização;
    Para cada mintermo negativo P faça
              N = ∅;
              T0 = intervalos de T que contém P;
              T1 = intervalos de T que não contém P;
              Para cada intervalo I de T0 faça
                  Elimine P de I e gere os sub-intervalos I'i;
                  Para cada I'i,
                      se I'i ∉ T1
                          então N = N ∪ {I'i};
              T = T1 ∪ N;
    Devolva T;
}
```

O algoritmo ISI.

O processamento sucessivo dos passos deste algoritmo pode ser visualizado através de uma árvore, onde os nós de um determinado nível são os intervalos resultantes após a execução do passo correspondente ao nível. Isto é, se consideramos a raíz no nível zero, então os nós no primeiro nível são os intervalos resultantes após a extração do ponto correspondente ao primeiro mintermo negativo; os nós no segundo nível são os intervalos correspondentes à extração do ponto correspondente ao segundo mintermo negativo e assim por diante. Esta dinâmica para a minimização da função $f = \sum m(0,1,4,5,6)$ é mostrada na figura 6.2. Geometricamente, este mesmo processo pode ser ilustrado pelos respectivos cubos no n-espaço, conforme mostra a figura 6.2.

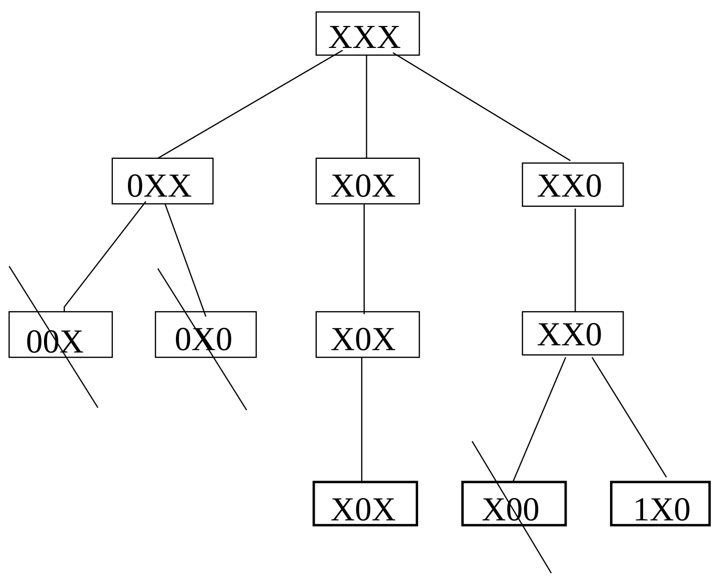


Figura 6.1: Expansão em árvore produzida pelo algoritmo ISI para $f = \sum m(0,1,4,5,6)$.

Como o resultado do algoritmo ISI corresponde ao conjunto de todos os implicantes primos, existe necessidade de selecionarmos os essenciais, da mesma forma que procedemos na segunda parte do algoritmo QM (ver capítulo 2). Quando nos referirmos ao algoritmo ISI daqui para frente, estaremos considerando o algoritmo descrito acima mais a segunda parte do algoritmo QM.

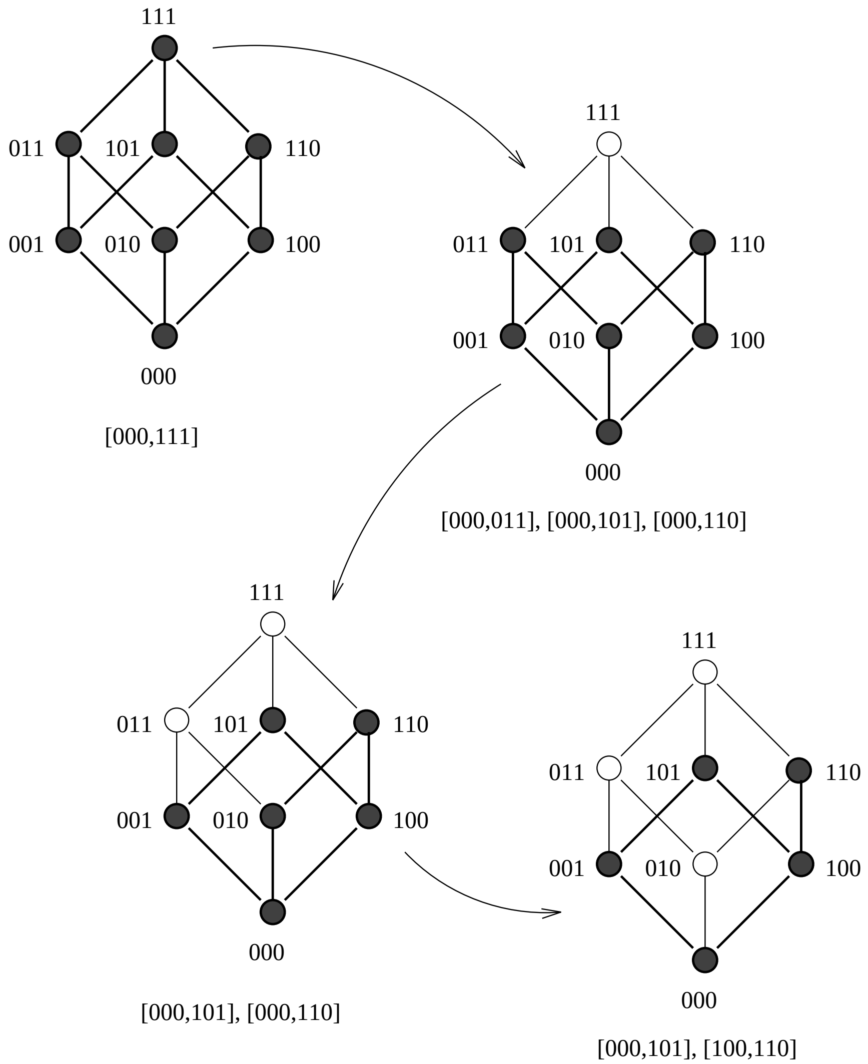


Figura 6.2: Visualização do efeito causado pelo algoritmo ISI no n-cubo para $f = \sum m(0, 1, 4, 5, 6)$.

6.1.2 Complexidade

Conforme ilustra a figura 6.1, o comportamento deste algoritmo pode ser visualizado através de um gráfico em forma de uma árvore. A profundidade da árvore é equivalente ao número de mintermos negativos. É interessante notar também que, quanto mais descemos o nível na árvore, menor é o grau dos nós e que, em termos de implementação, somente as folhas da árvore devem ser conservadas.

A expansão da árvore produzida pelo algoritmo depende da distribuição dos mintermos, o que torna difícil uma análise da complexidade do algoritmo. Portanto optamos por ilustrar a sua complexidade através de comparações com o algoritmo QM, o que é bem conhecido.

6.2 Quine-McCluskey $\times$ ISI

Para que uma função Booleana f com n variáveis seja completamente definida, devemos conhecer o seu valor em cada um dos possíveis 2^n pontos do seu domínio. Na prática é muito comum que o valor da função em alguns (ou vários) destes pontos seja desconhecido ou irrelevante. Este conjunto de pontos é denominado "don't care". Em outros termos, significa dizer que não interessa qual valor toma a função nestes pontos.

No algoritmo QM, estes pontos devem ser tratados como mintermos positivos, já que isto aumenta a possibilidade de combinações, para gerar cubos maiores, nas fases intermediárias do algoritmo. E, em conseqüência, a possibilidade dos implicantes primos serem cubos de dimensão maior também aumenta, o que implica menos cubos ou cubos maiores no conjunto de implicantes primos essenciais.

Note que na tabela de implicantes primos, utilizada para selecionar os implicantes primos essenciais na segunda parte do algoritmo, não há necessidade de incluirmos os "don't cares" nas colunas, uma vez que a função minimal resultante não precisa, necessariamente, tomar valor 1 nestes pontos.

Por outro lado, este procedimento de considerar todos os "don't cares" como mintermos positivos possui a desvantagem de influir diretamente no aumento de espaço e tempo necessários para o processamento, pois causa um aumento no número de combinações nas fases intermediárias do algoritmo, além da necessidade de maior espaço de memória para armazená-los.

A diferença significativa entre o algoritmo ISI e QM está exatamente no que diz respeito aos "don't cares". No caso do algoritmo ISI, todos os mintermos que não são negativos são considerados positivos (com excessão da segunda parte do algoritmo, no qual os intervalos que contém apenas "don't cares" são eliminados).

Em outras palavras, a primeira parte do algoritmo ISI depende dos mintermos negativos, enquanto que o QM depende dos mintermos positivos e "don't cares". O algoritmo ISI permite simplificar funções com muitas variáveis, desde que o número de "don't cares" seja relativamente grande. Isto não é possível com o algoritmo QM.

Apresentamos a seguir, uma série de tabelas que comparam o desempenho dos algoritmos QM e ISI para várias situações diferentes. As colunas "negativos", "positivos" e "d.c." indicam, respectivamente, o número de mintermos negativos, positivos e don't cares; a coluna "base" indica o tamanho da base (implicantes primos essenciais) obtida; o tempo é medido em horas(h), minutos(m) e segundos(s); "mem." indica a quantidade máxima de intervalos (cubos) que coexistiram simultaneamente durante o processo de minimização. O tempo quando medido em unidades menores que o segundo não é muito preciso, portanto para efeitos de análise, desprezamos este tempo arredondando aqueles maiores que $0.5s$ para $1s$ e os menores ou iguais a 0.5 para $0s$. O tempo indicado por ∞ representa tempo superior a 24 horas. O espaço ("mem.") indicado por ∞ indica que o espaço em memória disponível no equipamento não foi suficiente para realizar o respectivo teste.

Todos os experimentos foram realizados em um equipamento com processador Pentium, 82 Mhz. Os dados foram gerados aleatoriamente.

				ISI		QM	
negativos	positivos	d.c.	base	tempo	mem.	tempo	mem.
10	502	0	85	1s	234	1s	1,075
51	461	0	59	6s	898	40s	18,702
154	358	0	80	3s	399	2s	2,301
256	256	0	85	1s	234	1s	1,075
260	252	0	84	1s	221	2s	1,054
410	102	0	52	1s	96	1s	222
128	128	256	41	2s	415	2s	2,885
205	51	256	30	1s	331	1s	1,549
216	212	84	27	0s	136	1s	2,489
277	170	65	37	0s	137	0s	1,479
312	193	7	27	1s	241	0s	1,415
504	1	7	1	1s	207	0s	10

Tabela 6.1: Tabela comparativa para 9 variáveis.

Os valores observados na tabela 6.1 não oferecem evidências muito concretas acerca do tempo de processamento, mas está claro que o algoritmo QM utiliza muito mais espaço de memória (medido em número de intervalos) do que o ISI. Na primeira linha da tabela 6.2

				ISI		QM	
negativos	positivos	d.c.	base	tempo	mem.	tempo	mem.
21	1,003	0	38	3s	479	1m2s	22,542
51	973	0	59	6s	898	40s	18,702
102	922	0	88	7s	909	24s	13,910
205	818	0	132	9s	1,099	10s	8,032
307	717	0	155	12s	945	10s	5,459
512	512	0	159	8s	563	7s	2,414
819	205	0	97	1s	188	0s	493
922	102	0	57	1s	96	1s	207
256	256	512	79	1s	939	9s	6,770
410	102	512	54	4s	741	2s	3,711

Tabela 6.2: Tabela comparativa para 10 variáveis.

pode-se verificar uma grande diferença de tempo entre o ISI e o QM. Casos em que há poucos mintermos negativos são favoráveis ao ISI, enquanto os casos em que há muitos mintermos positivos são desfavoráveis ao QM.

				ISI		QM	
negativos	positivos	d.c.	base	tempo	mem.	tempo	mem.
205	1,843	0	148	1m46s	2,337	3m32s	36,744
359	1,689	0	198	1m3s	2,126	1m46s	24,213
655	1,393	0	275	59s	1,991	51s	12,071
1024	1,024	0	281	30s	1,191	23s	5,599
1689	359	0	174	4s	332	1s	843
208	688	1,152	96	46s	2,350	2m31s	36,443
425	688	935	135	50s	2,082	1m12s	21,666
754	359	935	112	19s	1,386	16s	11,384

Tabela 6.3: Tabela comparativa para 11 variáveis.

Na tabela 6.3 observamos que o tempo de processamento em vários casos aproxima-se ou supera os minutos, enquanto na tabela anterior, com 10 variáveis somente um caso passou de um minuto. Podemos observar um crescimento muito mais rápido do tempo de processamento e espaço ocupado relativamente ao crescimento do número de variáveis.

				ISI-0		QM	
negativos	positivos	d.c.	base	tempo	mem.	tempo	mem.
1035	3,061	0	430	7m	4,580	7m7s	38,524
2048	2,048	0	418	1m30s	1,942	1m22s	15,876
3061	1,035	0	411	26s	993	14s	3,069
712	724	2,660	90	24s	2,202	20m10s	89,824
835	601	2,660	82	21s	2,160	16m	80,682

Tabela 6.4: Tabela comparativa para 12 variáveis.

Na tabela 6.4, torna-se evidente que o algoritmo ISI é muito mais eficiente do que o QM nos casos para os quais o número de "don't cares" é relativamente grande.

				ISI-0		QM	
negativos	positivos	d.c.	base	tempo	mem.	tempo	mem.
16.384	16.384	0	1.768	6h46m15s	12,728	7h19m54s	213,101
42	18	32,708	6	29s	4,146	∞	-
565	16,384	15,819	220	35m2s	13,900	-	-
1.706	1,031	30,031	110	26m18s	21,389	∞	-
12.880	18	19,878	17	26m35s	12,094	-	-

Tabela 6.5: Tabela comparativa para 15 variáveis.

Na tabela 6.5, podemos verficar que o processamento em alguns casos levou horas. Observe novamente que quando o número de "don't cares" é grande, o ISI é estremamente eficiente (veja linha 2), porém não se pode generalizar (compare linhas 2 e 4). Na linha 4, referente ao QM, o tempo de processamento (não terminado) foi superior a 4 dias e meio.

Na tabela 6.6 apresentamos o desempenho do ISI para alguns casos com 16, 20, 25 e 39 variáveis. Não realizamos os testes com o QM para estes casos, pois estes levariam dias de processamento.

6.3 As Variantes do Algoritmo ISI

Verificamos que o algoritmo ISI elimina sucessivamente os mintermos negativos e expressa os mintermos restantes em termos de um conjunto de intervalos maximais. Estes intervalos maximais cobrem todos os mintermos positivos e também todos os "don't cares". Na segunda parte do algoritmo, todos os intervalos que cobrem apenas "don't cares" (isto é,

variáveis	negativos	positivos	d.c.	base	tempo	mem.
16	163	204	65,169	4	2s	550
16	177	193	65,166	3	2s	529
16	2,069	1,155	62,312	91	1h9m38s	39,440
20	285	306	1,047,985	2	6s	929
20	21	1052	1,047,503	25	8s	744
25	36	2,493	33,551,903	32	1h42m1s	32,880
25	472	446	33,553,514	3	32s	2,308
39	740	803	$2^{39} - 1,543$	3	9m18s	10,455

Tabela 6.6: Desempenho do ISI.

aqueles ques não contém nenhum mintermo positivo) são eliminados. Como conseqüência deste processamento, o resultado final consiste de um conjunto mínimo de intervalos maximais necessários para cobrir todos os mintermos positivos.

Uma observação decorrente deste fato é a possibilidade de uma pequena alteração na primeira parte do algoritmo ISI. Para isto, observe o seguinte fato : se um intervalo não contém nenhum mintermo positivo, então todos os seus sub-intervalos também não contêm. Logo, intervalos com esta propriedade podem ser eliminados após cada passo na primeira parte do algoritmo.

Esta observação levou-nos a sugerir três variantes do algoritmo ISI. Para evitar confusões entre eles, denominaremos de ISI-0 a versão original do algoritmo e de ISI-1, ISI-2 e ISI-3, respectivamente, as três variantes.

6.3.1 Algoritmo ISI-1

Nesta variante do ISI, ao final de cada passo da primeira parte do algoritmo, eliminamos todos os intervalos que não contém nenhum mintermo positivo. Isto corresponde a antecipar a filtragem que na versão original é realizada na segunda parte do algoritmo.

Com este procedimento, o tempo de processamento médio em relação ao ISI-0 diminui, pois o algoritmo manipula menos intervalos durante o processo. Outro fato bastante interesante é que o ISI-1, embora não produza os mesmos implicantes primos que o ISI-0 na primeira parte do algoritmo, produz os mesmos intervalos essenciais ao final do processo.

A seguir apresentamos o algoritmo ISI-1, em pseudo-código. Observe que apenas uma linha (marcada por *) foi alterada em relação à versão original do algoritmo.

Entrada: - Conjunto de mintermos positivos e negativos
- Intervalos de inicialização
Saída : - Conjunto de implicantes primos essenciais

{
    $\mathcal{T}$ = intervalos de inicialização;
    Para cada mintermo negativo P faça
        $\mathcal{N} = \emptyset$;
        $\mathcal{T}_0$ = intervalos de $\mathcal{T}$ que contém P;
        $\mathcal{T}_1$ = intervalos de $\mathcal{T}$ que não contém P;
        Para cada intervalo I de $\mathcal{T}_0$ faça
            Elimine P de I e gere os sub-intervalos I'_i;
            Para cada I'_i,
                se $I'_i \notin \mathcal{T}_1$ e I'_i contém algum mintermo positivo (*)
                    então $\mathcal{N} = \mathcal{N} \cup \{I'_i\}$;
        $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{N}$;
    Devolva $\mathcal{T}$;
}

Algoritmo ISI-1.

6.3.2 Algoritmo ISI-2

A segunda variante do ISI consiste em gerar, a cada passo, alguns subintervalos até que todos os mintermos positivos estejam cobertos, ou seja, quando se extrai um ponto de um intervalo de dimensão k, nem todos os k subintervalos de dimensão $k-1$ são gerados. Isto torna esta versão do algoritmo mais estável quanto a sua complexidade de espaço (que será limitado pelo número de exemplos positivos). Por outro lado, alguns dos intervalos que deixaram de ser gerados poderiam ser os intervalos essenciais. Em conseqüência, o resultado final poderá conter mais intervalos em relação ao resultado fornecido pelas variantes anteriores. Esta é uma versão que fornece um resultado sub-ótimo.

A seguir apresentamos o algoritmo ISI-2, em pseudo-código. Observe que as linhas marcadas por * representam as alterações em relação à versão original do algoritmo.

```
Entrada:  - Conjunto de mintermos positivos e negativos
          - Intervalos de inicialização
Saída :   - Conjunto de implicantes primos essenciais

{
    T = intervalos de inicialização;
    Para cada mintermo negativo P faça
             N = ∅;
             T0 = intervalos de T que contém P;
             T1 = intervalos de T que não contém P;
             cobertura-ok = falso;                          (*)
             Para cada intervalo I de T0 enquanto cobertura-ok = falso faça (*)
                 Enquanto cobertura-ok = falso e
                     não foi gerado todos os subintervalos de I      (*)
                     Gere o próximo subintervalo I'_i ;  (*)
                     Se I'_i cobre mintermo positivo não coberto por N ∪ T1 então (*)
                         N = N ∪ {I'_i};
                         se todos os mintermos positivos estão
                             cobertos pelos intervalos de N ∪ T1
                             então cobertura-ok = verdadeiro;              (*)
             T = T1 ∪ N;
    Devolva T;
}
```

Algoritmo ISI-2.

6.3.3 Algoritmo ISI-3

Na terceira variante do ISI, escolhe-se um subconjunto ótimo de intervalos necessários para cobrir todos os mintermos positivos ao final de cada passo (isto é, aplica-se o procedimento correspondente à segunda parte do algoritmo QM). Pela mesma razão do ISI-2, o resultado final do ISI-3 pode conter mais intervalos do que o ISI-1. A prática mostra que este número é menor do que na segunda variante (isto deve-se provavelmente ao fato de que a cada passo escolhemos um subconjunto ótimo naquele momento). Esta versão do algoritmo também fornece um resultado sub-ótimo.

Entrada: - Conjunto de mintermos positivos e negativos
- Intervalos de inicialização
Saída : - Conjunto de implicantes primos essenciais

{
    $\mathcal{T}$ = intervalos de inicialização
    Para cada mintermo negativo P faça
        $\mathcal{N} = \emptyset$;
        $\mathcal{T}_0$ = intervalos de $\mathcal{T}$ que contém P;
        $\mathcal{T}_1$ = intervalos de $\mathcal{T}$ que não contém P;
        Para cada intervalo I de $\mathcal{T}_0$ faça
            Elimine P de I e gere os sub-intervalos I'_i;
            Para cada I'_i,
                se $I'_i \notin \mathcal{T}_1$ e I'_i contém mintermo positivo (*)
                    então $\mathcal{N} = \mathcal{N} \cup \{I'_i\}$;
        $\mathcal{T}$ = subconjunto ótimo de $\mathcal{T}_1 \cup \mathcal{N}$
            que cobre todos os mintermos positivos; (*)
    Devolva $\mathcal{T}$;
}

Algoritmo ISI-3.

6.3.4 Análise Comparativa

As complexidades dos algoritmos ISI são difíceis de serem calculadas uma vez que o desempenho de cada um deles depende fortemente da distribuição dos mintermos positivos, mintermos negativos e "don't cares".

Na seqüência apresentamos várias tabelas que comparam o desempenho dos algoritmos ISI-0, ISI-1, ISI-2 e ISI-3. As observações feitas para as tabelas da seção anterior continuam valendo para estas tabelas. O sinal "?" presente em algumas tabelas indica um valor que não foi anotado. Optamos por apresentar estas tabelas, mesmo que incompletas, pois representam dados obtidos a partir de problemas reais (ver experimentos descritos no capítulo 10).

12 **variáveis**

Uma das primeiras características das variantes do ISI que podemos observar é que nos casos em que não há "don't cares" o ISI-2 e ISI-3 são ineficientes quando comparados com o ISI-0 e ISI-1 (veja as tabelas (c), (d) e (e) a seguir). O ISI-3 é, em particular, mais ineficiente pois, a cada passo, ele calcula um subconjunto ótimo que cobre todos os mintermos positivos.

a)

$$\begin{array}{lcll} \text{negativos} & = & 1,128 & (27.54\%) \\ \text{positivos} & = & 25 & (0.61\%) \\ \text{don't care} & = & 2,943 & (71.85\% \end{array}$$

Alg.	tempo	mem.	base
ISI-0	29s	2,154	8
ISI-1	1s	115	8
ISI-2	1s	14	9
ISI-3	1s	21	8

b)

$$\begin{array}{lcll} \text{negativos} & = & 24 & (0.59\%) \\ \text{positivos} & = & 1,129 & (27.56\%) \\ \text{don't care} & = & 2,943 & (71.85\% \end{array}$$

Alg.	tempo	mem.	base
ISI-0	1s	32	16
ISI-1	1s	32	16
ISI-2	1s	23	21
ISI-3	1s	24	18

c)

$$\begin{array}{lcll} \text{negativos} & = & 2,048 & (50\%) \\ \text{positivos} & = & 2,048 & (50\%) \\ \text{don't care} & = & 0 & \end{array}$$

Alg.	tempo	mem.	base
ISI-0	1m39s0	1,942	418
ISI-1	1m45s	1,942	418
ISI-2	12m45s	706	703
ISI-3	19m40s	496	491

d) negativos = $1,035$ (25.3%)
positivos = $3,061$ (74.7%)
don't care = 0

Alg.	tempo	mem.	base
ISI-0	7m	4,580	430
ISI-1	7m12s	4,580	430
ISI-2	14m2s	862	855
ISI-3	26m20s	588	576

e) negativos = $3,061$ (74.7%)
positivos = $1,035$ (25.3%)
don't care = 0

Alg.	tempo	mem.	base
ISI-0	26s	993	411
ISI-1	29s	991	411
ISI-2	6m40s	557	551
ISI-3	13m14s	487	480

15 **variáveis**

Na tabela (b) a seguir, o ISI-3 não conseguiu terminar o processamento por falta de espaço em memória. Na coluna *mem.* de todas as tabelas desta seção estamos indicando o número máximo de "folhas" na árvore gerado pelo algoritmo durante o processamento. Em geral, podemos observar que "mem." referente ao ISI-3 é menor que o referente a ISI-1, por exemplo. Entretanto, antes de gerar as folhas da árvore, o algoritmo ISI-3 constrói a tabela dos implicantes primos que consiste de uma matriz cujas colunas estão associadas aos mintermos positivos e as linhas são associadas aos intervalos. Esta é uma matriz que pode ocupar muito mais espaço em memória do que o espaço necessário para armazenar o número de intervalos indicado em *mem.*, e isto justifica por que o ISI-3 deparou com o problema de falta de espaço em memória.

a) negativos = 42 (0.13%)
positivos = 18 (0.05%)
don't care = $32,708$ (99.82%)

Alg.	tempo	mem.	base
ISI-0	29s	4,146	6
ISI-1	12s	1,820	6
ISI-2	2s	17	14
ISI-3	2s	29	9

b) negativos = $16,384$ (50%)
positivos = $16,384$ (50%)
don't care = 0

Alg.	tempo	mem.	base
ISI-0	6h46m14s	12,728	1768
ISI-1	6h56m6s	12,728	1768
ISI-2	97h30m16s	4,608	4601
ISI-3	-	∞	-

c) negativos = $12,880$ (39.31%)
positivos = 18 (0.05%)
don't care = $19,870$ (60.64%)

Alg.	tempo	mem.	base
ISI-0	26m35s	12,094	17
ISI-1	8s	166	17
ISI-2	3s	19	18
ISI-3	3s	36	17

d)
$$\begin{array}{lcll} \text{negativos} & = & 565 & (1.7\%) \\ \text{positivos} & = & 16,384 & (50\%) \\ \text{don't care} & = & 15,819 & (48.3\%) \end{array}$$

Alg.	tempo	mem.	base
ISI-0	35m3s	13,900	220
ISI-1	37m1s	13,900	220
ISI-2	44m10s	1.004	992
ISI-3	34m53s	394	347

16 **variáveis**

Em todas as tabelas observa-se que a base obtida pelos algoritmos ISI-0 e ISI-1 possui o mesmo tamanho. Observa-se também que o ISI-3 tem um resultado melhor do que o ISI-2 em todos os casos. A base obtida pelo ISI-3 tem tamanho próximo da base ótima (obtida pelo ISI-0 e ISI-1) em vários casos, enquanto a base obtida pelo ISI-2 tem tamanho muito aquém do ótimo em vários casos.

a)
$$\begin{array}{lcll} \text{negativos} & = & 2,069 & (3.16\%) \\ \text{positivos} & = & 1,155 & (1.76\%) \\ \text{don't care} & = & 62,312 & (95.08\%) \end{array}$$

Alg.	tempo	mem.	base
ISI-0	1h9m38s	39,440	91
ISI-1	49m34s	29,108	91
ISI-2	16s	180	174
ISI-3	39s	174	99

b)
$$\begin{array}{lcll} \text{negativos} & = & 177 & (0.27\%) \\ \text{positivos} & = & 193 & (0.29\%) \\ \text{don't care} & = & 65,166 & (99.44\%) \end{array}$$

Alg.	tempo	mem.	base
ISI-0	2s	529	3
ISI-1	2s	528	3
ISI-2	1s	11	10
ISI-3	1s	18	3

c)
$$\begin{array}{lcll} \text{negativos} & = & 163 & (0.25\%) \\ \text{positivos} & = & 204 & (0.31\%) \\ \text{don't care} & = & 65,169 & (99.44\%) \end{array}$$

Alg.	tempo	mem.	base
ISI-0	2s	12,094	17
ISI-1	1s	166	17
ISI-2	1s	19	18
ISI-3	1s	36	17

20 **variáveis**

Com 20 variáveis o ISI-0 torna-se ineficiente para vários casos. Observe as tabelas (c) e (d) abaixo.

a)
$$\begin{array}{lcll} \text{negativos} & = & 21 & (0.002\%) \\ \text{positivos} & = & 1,052 & (0.100\%) \\ \text{don't care} & = & 1,047,503 & (99.898\%) \end{array}$$

Alg.	tempo	mem.	base
ISI-0	8.4s	744	25
ISI-1	7.9s	743	25
ISI-2	1s	61	55
ISI-3	2.1s	170	27

b)

negativos	=	$1,020$	(0.097%)
positivos	=	53	(0.005%)
don't care	=	$1,047,503$	(99.898%)

Alg.	tempo	mem.	base
ISI-0	-	-	-
ISI-1	-	-	-
ISI-2	0.6s	44	43
ISI-3	0.9s	44	27

c)

negativos	=	$1,235$	(0.1178)
positivos	=	10	(0.0009)
don't care	=	$1,047,331$	(99.8813)

Alg.	tempo	mem.	base
ISI-0	-	∞	-
ISI-1	3m7s	2,247	10
ISI-2	1.5s	11	10
ISI-3	0.7s	23	10

d)

negativos	=	$1,803$	(0.1719%)
positivos	=	6	(0.0006%)
don't care	=	$1,046,767$	(99.8275%)

Alg.	tempo	mem.	base
ISI-0	128h29m3s	258,251	6
ISI-1	49.5s	1,056	6
ISI-2	0.7s	8	6
ISI-3	0.6s	21	6

e)

negativos	=	285
positivos	=	306
don't care	=	$1,047,985$

Alg.	tempo	mem.	base
ISI-0	7.9s	1,095	3
ISI-1	4.7s	1,080	3
ISI-2	0.9s	11	10
ISI-3	0.5s	22	3

25 variáveis

Para 25 variáveis o ISI-1 também torna-se ineficiente em vários casos.

a)

negativos	=	$2,493$	(0.0074%)
positivos	=	36	(0.0001%)
don't care	=	$33,551,903$	(99.9925%)

Alg.	tempo	mem.	base
ISI-0	-	-	-
ISI-1	-	-	-
ISI-2	0.9s	36	34
ISI-3	1.4s	73	27

b)

negativos	=	$1,451$	(0.0043%)
positivos	=	57	(0.0002%)
don't care	=	$33,552,924$	(99.9955%)

Alg.	tempo	mem.	base
ISI-0	-	-	-
ISI-1	-	-	-
ISI-2	0.6s	45	41
ISI-3	1.0s	53	29

c) negativos = 472 (0.0014)
positivos = 446 (0.0013)
don't care = 33,553,514 (99.9973)

Alg.	tempo	mem.	base
ISI-0	32.5s	2,308	3
ISI-1	19.8s	2,250	3
ISI-2	1.2s	11	10
ISI-3	0.8s	27	3

d) negativos = 36 (0.0001%)
positivos = 2,493 (0.0074%)
don't care = 33,551,903 (99.9925%)

Alg.	tempo	mem.	base
ISI-0	1h42m1s	32,880	32
ISI-1	1h41m41s	32,872	32
ISI-2	4.5s	113	111
ISI-3	12s	150	56

e) negativos = 21,336
positivos = 2,337

Alg.	tempo	mem.	base
ISI-3	20m53s	?	416

39 **variáveis**

Aqui queremos mostrar que alguns casos ainda podem ser resolvidos pelos algoritmos ISI-0 e ISI-1, mesmo que o número de variáveis seja grande.

a) negativos = 2,341
positivos = 55

Alg.	tempo	mem.	base
ISI-0	-	-	-
ISI-1	-	-	-
ISI-2	1,3s	45	32
ISI-3	1,3s	86	20

b) negativos = 2,228
positivos = 196

Alg.	tempo	mem.	base
ISI-0	-	-	-
ISI-1	-	-	-
ISI-2	1.4s	66	58
ISI-3	2.5s	130	17

c) negativos = 740
positivos = 803

Alg.	tempo	mem.	base
ISI-0	9m18s	10,455	3
ISI-1	5m42s	2,250	3
ISI-2	0.7s	9	8
ISI-3	0.7s	41	3

49 **variáveis**

Para esta quantidade de variáveis, o tempo gasto por ISI-0 e ISI-1 é extremamente alto. Por isso não realizamos testes com essas duas versões do ISI. Na tabela (b) abaixo pode-se verificar a diferença dos resultados fornecidos por ISI-2 e ISI-3. O tamanho da base gerado por ISI-2 tende a ser mais distante do ótimo quanto maior forem os mintermos negativos.

a) $\begin{array}{lcl} \text{negativos} & = & 2,338 \\ \text{positivos} & = & 150 \end{array}$

Alg.	tempo	mem.	base
ISI-2	2s	66	55
ISI-3	2.5s	118	21

b) $\begin{array}{lcl} \text{negativos} & = & 56,191 \\ \text{positivos} & = & 4,239 \end{array}$

Alg.	tempo	mem.	base
ISI-2	1h46m15s	?	1,348
ISI-3	24m31s	?	284

81 **variáveis**

Aqui observamos que alguns casos com 81 variáveis podem ser resolvidos em “poucas” horas ou até alguns segundos de processamento. Isto indica que outros casos com maior número de variáveis provavelmente poderão também ser resolvidos.

a) $\begin{array}{lcl} \text{negativos} & = & 2,545 \\ \text{positivos} & = & 303 \end{array}$

Alg.	tempo	mem.	base
ISI-2	3.3s	69	61
ISI-3	5.9s	163	15

b) $\begin{array}{lcl} \text{negativos} & = & 67,119 \\ \text{positivos} & = & 7,823 \end{array}$

Alg.	tempo	mem.	base
ISI-2	14h42m39s	?	2,798
ISI-3	3h44m43s	?	551

Nas tabelas que comparam o algoritmo ISI-0 com o algoritmo QM podemos verificar que quando a quantidade de “don't care” é grande, em geral o algoritmo ISI-0 é muito mais rápido que o algoritmo QM. O algoritmo QM mostra-se melhor quando o número de mintermos negativos é grande. Nos casos em que 50% dos mintermos são negativos e outros 50% são positivos, aparentemente o desempenho do QM é melhor. No entanto, para o caso de 15 variáveis não se pode afirmar a mesma coisa. Independentemente destes fatos, em geral o número máximo de intervalos gerados pelo QM durante o processamento é superior ao gerado pelo ISI-0. Isto leva-nos a concluir que se considerarmos um número crescente de variáveis, o algoritmo QM esbarrará primeiro no limite de memória disponível.

O algoritmo ISI-0 pode ser utilizado para casos nos quais o número de variáveis é grande, desde que o número de mintermos negativos e positivos seja relativamente pequeno. Se observarmos que na prática dificilmente encontraremos situações nos quais o valor da função Booleana está definido para todos os possíveis 2^n mintermos e sim, em geral para poucos deles, o algoritmo ISI-0 representa uma alternativa viável e eficiente.

Apesar destas características do ISI-0, os testes realizados mostram que os casos que envolvem alguns milhares de exemplos, com número de variáveis em torno de 20, leva algumas horas de processamento.

Em vários destes casos as variantes do ISI mostram-se bastante interessantes. Nos casos em que há poucos mintermos positivos, a variante ISI-1 mostra-se muito mais eficiente que ISI-0. Já no caso inverso a este, pode-se dizer que os dois são praticamente equivalentes.

Se estamos interessados em obter os implicantes primos essenciais e não todos os implicantes primos, o ISI-1 é equivalente ao ISI-0. Estas duas versões fornecem um resultado ótimo. No entanto, para vários casos envolvendo 20 ou mais variáveis, elas são praticamente inviáveis, exigindo dias de processamento.

Para estes casos, embora não forneça um resultado ótimo mas apenas sub-ótimo, as variantes ISI-2 e ISI-3 são alternativas bastante eficientes.

Em particular, o ISI-3 fornece um resultado melhor que o ISI-2. O resultado de ambos tende a ser pior, relativamente ao ótimo, quanto maior forem o número de exemplos negativos (pois a cada exemplo negativo extraído do cubo, vários "don't cares" são eliminados também). Este comportamento é mais acentuado no caso do ISI-2. Por outro lado, vários casos que não podem ou levam horas para serem resolvidos pelas variantes ISI-0 e ISI-1 podem ser resolvidos em questão de segundos pelo ISI-3 e ISI-2.

A vantagem do ISI-2 em relação aos demais é que o espaço de memória necessário é constante, no sentido de que é equivalente ao número de mintermos positivos (uma vez que no pior caso é necessário no máximo um intervalo para cobrir cada um dos mintermos positivos).

Verificamos que em geral o resultado do ISI-3 é melhor que o do ISI-2. Em algumas situações é também mais rápido. No entanto, uma vez que ele repete o processo de escolha do subconjunto ótimo, torna-se ineficiente quando existem muitos mintermos positivos e negativos (pois um grande número de mintermos negativos faz com que o número de intervalos gerados fique maior, e um grande número de mintermos positivos faz com que o número de intervalos necessários para cobrí-los seja relativamente maior também, e o processo de escolha de um subconjunto ótimo é um processo demorado quando existem muitos mintermos positivos e muitos intervalos).

A conclusão a que chegamos é que os algoritmos ISI representam uma alternativa viável para resolver muitos problemas de minimização de funções Booleanas que não são resolvidos pelos algoritmos clássicos. Além disso, observamos que todas as variantes do ISI são algoritmos consistentes com os mintermos positivos e negativos.

Capítulo 7

Geração Automática de Operadores Morfológicos

Neste capítulo estaremos interessados em modelar um sistema para programação automática de MMach's binárias, tendo como base os resultados teóricos apresentados nos capítulos anteriores (veja também [BTdST95]).

7.1 Considerações Preliminares

Conforme mencionamos nos capítulos anteriores, para programar uma MMach (para que esta resolva problemas de processamento de imagens), precisamos encontrar uma frase da *LM* que corresponde à solução do problema considerado.

Entretanto, escrever uma frase adequada da *LM* não é uma tarefa simples e requer conhecimentos específicos e profundos em *MM*. Em geral esta tarefa é realizada empiricamente, através de tentativas, o que nem sempre produz uma solução adequada.

O Teorema da Decomposição Canônica (Teorema 3.2.1) estabelece uma metodologia bem definida para a programação automática de MMach's, que pode ser definida pela seguinte seqüência de operações :

- Determinar o núcleo do operador desejado;
- Calcular os intervalos maximais do núcleo, isto é, a base do operador ;
- Escrever o operador como uma união de operadores sup-geradores, cujos elementos estruturantes (A, B) que os caracterizam correspondem aos extremos esquerdo e direito de cada intervalo da base.

Embora esta metodologia seja bem definida, a sua aplicação na prática não é em geral possível, pois se consideramos uma janela W com n pontos, existem 2^n possíveis elementos que pertencem ou não ao núcleo. Para uma janela relativamente pequena, por exemplo $n = 25$, o número de elementos que precisaríamos analisar relativamente ao fato de pertencerem ou não ao núcleo é $2^{25} = 33,554,432$.

Por outro lado, a programação automática de MMach's parece estar diretamente relacionada com a automatização destas tarefas, uma vez que elas estão bem definidas e são suficientes para caracterizar a classe de operadores i.t. e localmente definidos por uma janela.

Consideremos inicialmente o seguinte problema. Seja $(\boldsymbol{X}, \boldsymbol{Y})$ um par de conjuntos aleatórios que modelam as imagens antes e depois da transformação desejada. Queremos encontrar um operador ψ tal que, dada uma realização (X, Y) de $(\boldsymbol{X}, \boldsymbol{Y})$, $\psi(X)$ seja "uma boa aproximação" de Y.

Se $\boldsymbol{X}$ e $\boldsymbol{Y}$ forem estritamente estacionários, então eles podem ser caracterizados por uma janela W (veja teorema 4.1.1). Neste caso, podemos reformular a questão : estamos interessados em encontrar um operador i.t. e localmente definido pela janela W tal que $\psi(\boldsymbol{X})$ seja uma boa aproximação de $\boldsymbol{Y}$.

Vamos analisar primeiramente o caso em que existe um operador i.t. localmente definido por uma janela W tal que $\boldsymbol{Y} = \psi(\boldsymbol{X})$ (isto é, dadas realizações (X_1, Y_1) e (X_2, Y_2) de $(\boldsymbol{X}, \boldsymbol{Y})$, $X_1 = X_2 \Longrightarrow Y_1 = Y_2$). Nestas condições, se considerarmos todas as possíveis realizações de $(\boldsymbol{X}_W, \boldsymbol{Y}_W)$, pode-se calcular o núcleo de ψ e, conseqüentemente, a sua base através de um processo de minimização lógica.

Embora em teoria este seja um método ideal, na prática existem obstáculos que o tornam pouco viável. Um destes obstáculos refere-se à questão de como seriam gerados todas as possíveis realizações de $(\boldsymbol{X}_W, \boldsymbol{Y}_W)$ e o outro refere-se a quantidade de recursos computacionais necessários para manipular todos estes dados.

Os casos em que existem realizações (X_1, Y_1) e (X_2, Y_2) de $(\boldsymbol{X}, \boldsymbol{Y})$ tal que $X_1 = X_2$ mas $Y_1 \neq Y_2$ são aqueles para os quais não existe um operador i.t. e localmente definido por W tal que $\boldsymbol{Y} = \psi(\boldsymbol{X})$. Como estamos interessados em um operador ψ i.t. e localmente definido, estes casos apresentam uma questão adicional que envolve uma tomada de decisão : se (X, Y_1) e (X, Y_2) são duas possíveis realizações de $(\boldsymbol{X}, \boldsymbol{Y})$, a qual delas ψ deve se "ajustar" ? Isto é, deve-se escolher ψ tal que $\psi(X) = Y_1$ ou $\psi(X) = Y_2$?

A abordagem natural para contornar estes obstáculos é a inserção do problema no contexto estatístico. A idéia básica é estimar uma distribuição de probabilidade de $(\boldsymbol{X}, \boldsymbol{Y})$ a partir de algumas realizações e usar algum critério estatístico para tomar a decisão mencionada acima.

O trabalho de Dougherty (veja capítulo 4) corresponde exatamente a esta abordagem. O critério de decisão utilizado por Dougherty é aquele que minimiza o MAE (erro absoluto médio) segundo a distribuição de probabilidade estimada.

Um ponto delicado nesta abordagem é que não sabemos determinar a "qualidade" do operador estimado, pelo fato deste ter sido obtido a partir de uma distribuição de probabilidade estimada. O modelo de aprendizado PAC, apresentado no capítulo 5, oferece bases teóricas sólidas para formalizar e tratar esta questão.

Motivado por esse fato, apresentaremos nesta seção um método baseado no modelo de aprendizado PAC, objetivando a construção de um sistema computacional para programação automática de máquinas morfológicas binárias.

Do ponto de vista de um sistema computacional para programação automática de MMach's, as realizações de $(\boldsymbol{X}, \boldsymbol{Y})$ podem ser entendidas como exemplos que o usuário fornece ao sistema para indicar a operação que deseja realizar. Estes exemplos, representando respectivamente as imagens anterior e posterior à transformação desejada, podem ser gerados manualmente pelo usuário, por exemplo através de um editor de imagens, e serão denominados **imagens de treinamento**. As figuras 7.1 e 7.2 mostram alguns exemplos de imagens de treinamento. Na figura 7.3 esquematizamos a estrutura geral do

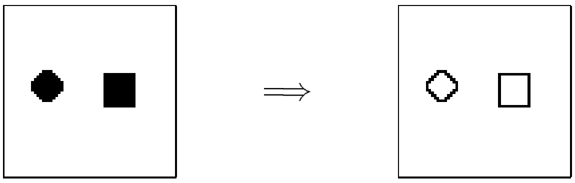


Figura 7.1: Extração de bordas.

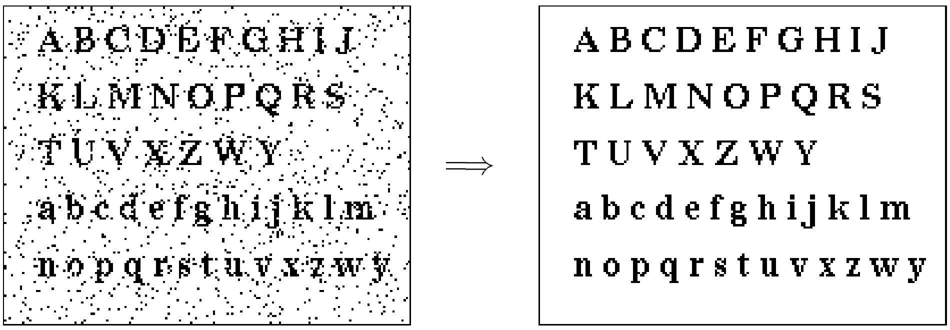


Figura 7.2: Filtragem de ruídos.

sistema que estamos propondo. Os elementos X e Y representam imagens que expressam o operador desejado (isto é, são realizações do conjunto aleatório $(\boldsymbol{X}, \boldsymbol{Y})$ que modela o domínio das imagens); "treinamento" refere-se a um processo no qual as informações contidas nas imagens X e Y serão convertidas para uma frase da LM, conforme descreveremos a seguir.

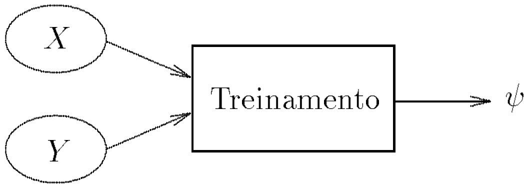


Figura 7.3: Modelo para programação automática de MMach's.

7.2 Especificação do Sistema

Vamos considerar as seguintes hipóteses :

1. as imagens a serem transformadas são realizações de um conjunto aleatório binário, denotado por $\boldsymbol{X}$;

2. as imagens correspondentes à transformação ideal de $\boldsymbol{X}$ são também realizações de um outro conjunto aleatório binário, denotado por $\boldsymbol{Y}$;

3. os conjuntos aleatórios que modelam as imagens são, cada um deles, estritamente estacionários e são também conjuntamente estacionários (isto é, $(\boldsymbol{X}, \boldsymbol{Y})$ é estritamente estacionário).

Sob a hipótese de estacionaridade estrita, podemos caracterizar um conjunto aleatório através de uma janela W. Logo, um par de conjuntos aleatórios $(\boldsymbol{X}, \boldsymbol{Y})$ estritamente estacionários, ou equivalentemente $(\boldsymbol{X}_W, \boldsymbol{Y}_W)$, pode ser caracterizado por um operador i.t. e localmente definido pela janela W.

O sistema será "treinado" para "aprender" um operador ψ i.t e localmente definido pela janela W a partir de imagens de treinamento (X, Y) fornecidos pelo usuário, segundo um critério de decisão que minimiza a perda especificada por uma função de perda l, também especificado pelo usuário (isto é, o sistema gera um operador de menor risco relativamente a l, conforme definição apresentada na página 72, segundo a distribuição de probabilidade conjunta estimada a partir de (X, Y)).

O processo do sistema envolve uma primeira etapa que consiste da estimação da distribuição de probabilidade conjunta sobre $(\boldsymbol{X}_W, \boldsymbol{Y}_W)$. Esta etapa será denominada "coleta de exemplos". A segunda etapa do processo refere-se a decisão já mencionada anteriormente. A terceira é a etapa na qual o sistema gera uma função Booleana (ou equivalentemente, a base de um operador i.t. e localmente definido) que é consistente com a decisão tomada na segunda etapa.

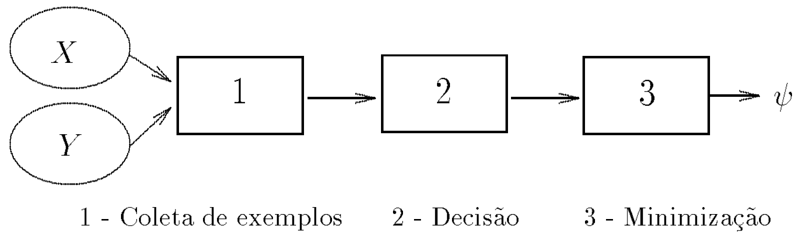


Figura 7.4: As três partes do sistema.

A figura 7.4 ilustra a estrutura do sistema com as três etapas :

A seguir, descrevemos cada uma das etapas detalhadamente.

7.2.1 Coleta de Exemplos

Seja (X, Y) um par de imagens de treinamento e seja $\mathcal{C}(X_W) = \{X \cap W + z : z \in E\}$. Os elementos de $\mathcal{C}(X_W)$ são denominados configurações e são obtidos transladando-se a janela W de modo que o seu centro fique sobre cada ponto de E, conforme mostra a figura 7.5.

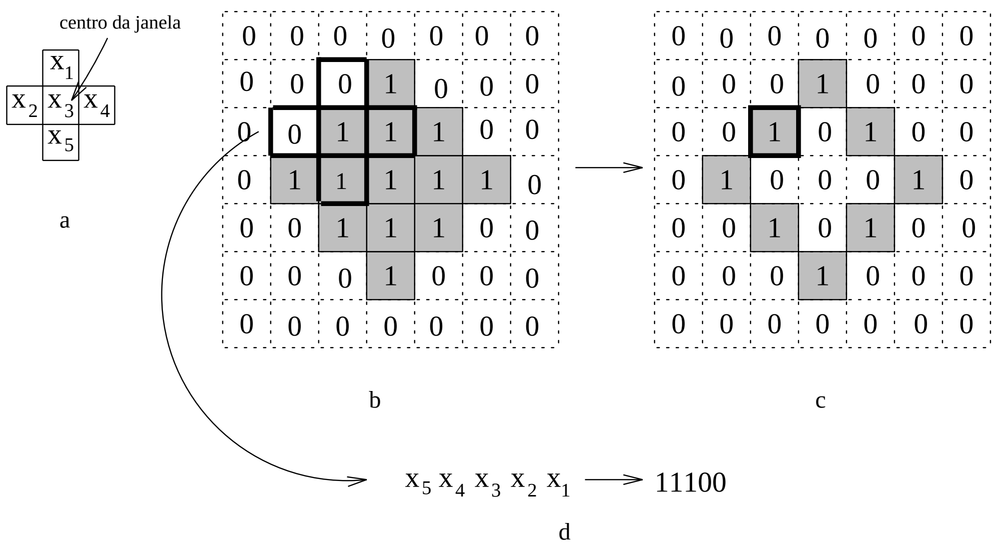


Figura 7.5: a) janela; b) e c) imagens de treinamento; d) uma configuração.

Cada elemento de $\mathcal{C}(X_W)$ pode ser descrito através de seqüências de bits. Por exemplo, a configuração observada pela janela W na figura 7.5 pode ser representada pela seqüência 11100. Para cada uma das configurações de $\mathcal{C}(X_W)$ encontra-se associado um valor Y_z igual a 0 ou 1 na segunda imagem. Ao se percorrer toda a imagem X, para todo z, obtém-se uma tabela consistindo de três colunas : a configuração, a freqüência com que ela ocorre com valor associado igual a 0 (f_0) e a freqüência com que ela ocorre com valor associado igual a 1 (f_1). Esta tabela define uma distribuição de probabilidade conjunta P sobre $X_W \times \{0,1\}$. As configurações tais que $f_0 \neq 0$ e $f_1 \neq 0$ serão denominadas configurações ou **exemplos contraditórios**.

A coleta de exemplos sobre as imagens da figura 7.5 resulta na tabela de exemplos especificada pela tabela 7.1.

Configuração	f_0	f_1
11100	0	1
00000	3	0
11000	2	0
10010	2	0
11111	6	0
10100	0	1
10110	0	1
01100	0	1
00110	0	1
01001	2	0
01101	0	1
00111	0	1
00011	2	0
00101	0	1

Tabela 7.1: Tabela de exemplos.

7.2.2 Decisão ótima

Baseado na distribuição P determinada pela tabela de exemplos, estamos interessados em encontrar um operador ψ, i.t. e localmente definido por W, que melhor represente a transformação caracterizada pelo par $(\boldsymbol{X}_W, \boldsymbol{Y}_W)$. Observe que, se ψ é i.t., então para qualquer $z \in E$,

$$\begin{aligned} z \in \psi(X \cap W + z) &\iff o \in \psi(X \cap W + z) - z \\ &\iff o \in \psi((X \cap W + z) - z) \end{aligned}$$

$$\begin{aligned} &\iff o \in \psi(X \cap W) \\ &\iff o \in \psi(X_W). \end{aligned}$$

Portanto, $X_W \in \mathcal{K}_W(\psi) \iff z \in \psi(X \cap W + z)$, e $z \in \psi(X \cap W + z)$ se Y_z é igual a 1.

Portanto, o núcleo de ψ pode ser estimado a partir das configurações obtidas no processo de coleta de exemplos.

Vamos inserir o problema no contexto de aprendizado. Isto é, estamos interessados em encontrar uma decisão (função Booleana) h de risco mínimo relativamente a uma função de perda l. Para cada configuração x da tabela de exemplos devemos decidir se $h(x) = 1$ ou $h(x) = 0$. Esta decisão é realizada segundo um critério de decisão que minimiza a perda caracterizada por l.

Se associarmos uma variável Booleana a cada um dos pontos da janela W, podemos encarar cada uma das configurações como mintermos de uma função Booleana. O processo de decisão resulta em uma segunda tabela, denominada **tabela de mintermos**, consistindo de duas colunas : uma coluna com os mintermos e outra com o valor de h para cada um dos mintermos.

Por exemplo, no caso da tabela de exemplos acima, se a função de perda é dada por $l(h(x), b) = |h(x) - b|$, onde (x, b) é um elemento de $X_W \times \{0, 1\}$, então deve-se fazer $h(x) = 1$ se $b = 1$ e $h(x) = 0$ se $b = 0$. Este processo origina a tabela de mintermos descrita pela tabela 7.2.

Configuração	h
11100	1
00000	0
11000	0
10010	0
11111	0
10100	1
10110	1
01100	1
00110	1
01001	0
01101	1
00111	1
00011	0
00101	1

Tabela 7.2: Tabela de mintermos obtida de uma tabela de exemplos.

Os mintermos da função Booleana h correspondem exatamente às configurações que fazem parte do núcleo estimado de ψ.

7.2.3 Minimização

Uma vez realizada a decisão ótima para a amostra considerada, precisamos gerar a função correspondente a esta decisão. Vamos considerar que os mintermos correspondentes às configurações não observadas são "don't cares" e aplicaremos o algoritmo ISI , que é um algoritmo de aprendizado consistente, sobre estes mintermos. O algoritmo ISI gera uma função Booleana f_ψ na FND que é consistente com a tabela de mintermos. Os termos da função Booleana f_ψ correspondem aos intervalos da base do operador ψ aprendido. A conversão para a notação de intervalos é trivial, conforme regra apresentada no capítulo 3.

Observe que o operador aprendido é um operador ótimo em relação à distribuição de probabilidade P, segundo a função de perda l.

7.2.4 Aplicação

Seja $\mathcal{B}(\psi) = \{[A_i, B_i] : i \in I\}$ a base do operador ψ aprendido e seja X uma realização de $\boldsymbol{X}$. Então, conforme o teorema 3.2.2 a transformação $\psi(X)$ pode ser obtida calculando-se :

$$\psi(X) = \bigcup\{\lambda^W_{(A,B)}(X) : [A, B] \in \mathcal{B}(\psi)\}$$

Denominaremos a estrutura definida neste capítulo como **estrutura básica do sistema** (figura 7.6). Observe que para realizar o treinamento, o usuário deve fornecer as imagens de treinamento, uma janela de observação W e um critério de decisão que minimiza a perda. Após o treinamento, o sistema devolve a base de um operador i.t. e localmente definido pela janela W. A base obtida poderá ser aplicada sobre qualquer realização X do conjunto aleatório $\boldsymbol{X}$.

7.3 Considerações de Aspecto Prático

Do ponto de vista prático, um usuário deste sistema poderá deparar com as seguintes questões :

- qual deve ser a janela a ser utilizada ?
- quantas imagens de treinamento são necessárias ?

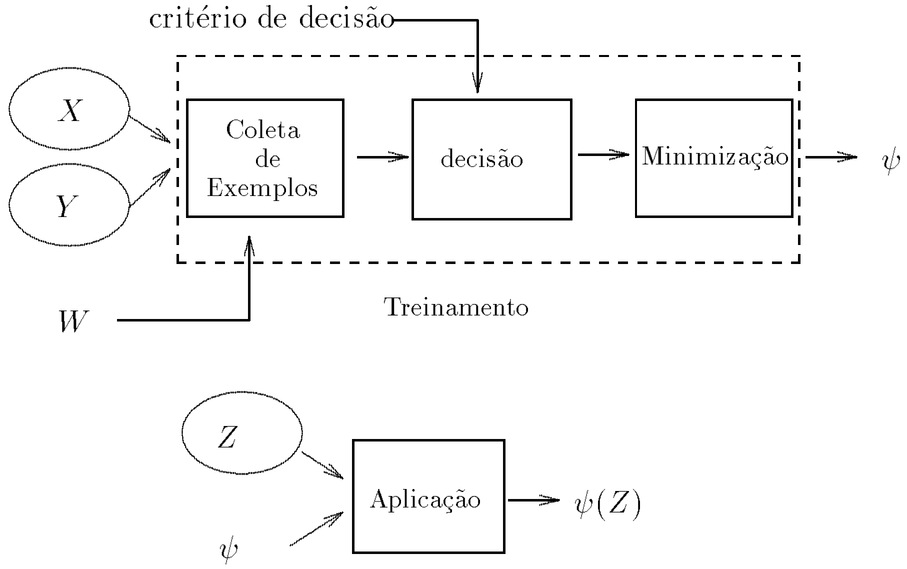


Figura 7.6: Estrutura básica do sistema.

- qual critério de decisão deve ser utilizado ?

Todas estas questões ainda não possuem resposta. Quanto ao tamanho da janela a ser utilizada, sugerimos dois critérios :

- lembrando que a janela delimita uma vizinhança ao redor de um ponto qualquer da imagem, cujas características serão importantes para definir o valor da transformação neste ponto, deve-se tentar delimitar esta vizinhaça levando em consideração as formas e o tamanho dos objetos presentes na imagem que se deseja transformar. A janela escolhida deve ser maior ou igual a esta vizinhança.

- Outra sugestão é a análise do conjunto $\mathcal{C}(X_W)$ para diferentes tamanhos de janela. Se os elementos de $\mathcal{C}(X_W)$ não forem contraditórios, então o aumento da janela não introduzirá contradições (ver janela mínima no capítulo 3). Por outro lado, o aumento do tamanho da janela tende a diminuir o número de elementos contraditórios em $\mathcal{C}(X_W)$.

  Para realizar esta análise, deve-se repetir a coleta de exemplos sobre as imagens de treinamento (X, Y) para janelas sucessivamente maiores, tomando-se o cuidado para que cada janela contenha a janela anterior. Os dados devem ser representados em um gráfico, cuja abscissa corresponde ao tamanho das janelas e a ordenada corresponde à porcentagem de elementos contraditórios em $\mathcal{C}(X_W)$ para a janela considerada.

Quando a curva gerada por este processo atingir uma certa estabilidade (isto é, quando a diferença relativa do valor da curva entre dois pontos consecutivos da abscissa tornar-se muito pequena) ou se a ordenada atingir um valor muito pequeno, o aumento da janela não irá acrescentar maiores informações ao conjunto de exemplos. Portanto, o tamanho da janela no ponto em que a curva se estabiliza pode ser considerado um bom parâmetro para a escolha do tamanho da janela a ser utilizada para o treinamento.

Apesar de ser útil para estimar o tamanho da janela, não existe nenhuma garantia de que este método funciona. Além disso, janelas muito grandes comprometem o treinamento (ver capítulo 10).

A quantidade de imagens necessárias para o treinamento está relacionada com a precisão da distribuição de probabilidade conjunta a ser estimada a partir deles. Intuitivamente, quanto maior o número de imagens, melhor será esta estimação e vice-versa.

No caso em que $\mathcal{C}(\boldsymbol{X}_W)$ não apresenta elementos contraditórios, recaímos no modelo PAC básico. E neste caso, um limitante superior para o número de exemplos necessários para se obter uma precisão (δ, ϵ) (no sentido PAC apresentado no capítulo 5) é dado por :

$$m_0 = \frac{1}{\epsilon} ln \frac{|H|}{\delta}$$

onde $H = 2^{2^{|W|}}$.

Quanto ao caso mais genérico, no qual existem elementos contraditórios em $\mathcal{C}(\boldsymbol{X}_W)$, este limitante não está bem definido[1].

Quanto a decisão associada aos exemplos contraditórios na fase de decisão ótima, a principal pergunta está associado ao critério que deve ser utilizado para decidir se um dado exemplo contraditório é positivo ou negativo. O usuário deve escolher uma função de perda e fornecer um critério de decisão que minimiza o risco. Mas como são estas funções de perda ?

O caso mais trivial é aquele no qual o domínio de exemplos não apresenta contradições. Neste caso, uma função de perda pode ser dada por :

$$l_h(x,b) = \begin{cases} 1 & \text{se } h(x) \neq b \\ 0 & \text{se } h(x) = b \end{cases}$$

[1] Pode-se encontrar um estudo relativo a este valor em [Hau92]. No entanto, preferimos omití-lo deste texto pois o estudo envolve vários conceitos complexos, além de tratar casos mais genéricos do que o tratado neste texto.

e o critério de decisão que minimiza a perda para (x, b) é dado por :

$$h(x) = \begin{cases} 1 & \text{se } b = 1 \\ 0 & \text{se } b = 0 \end{cases}$$

No caso de problemas de restauração existem exemplos contraditórios no domínio. Portanto não existe um operador i.t. (isto é, uma função Booleana) que realiza uma restauração 100% correta. Admitindo que o operador não seja perfeito neste sentido, queremos encontrar dentre todos eles, aquele que "erra menos". A função de perda neste caso é dada por :

$$l_h(x, b) = |h(x) - b|.$$

O critério que minimiza a perda é a decisão ótima de Bayes, dada por :

$$h(x) = \begin{cases} 1 & \text{se } p(b = 1/x) \geq 0.5 \\ 0 & \text{se } p(b = 1/x) < 0.5 \end{cases}$$

Destacamos quatro pontos básicos que diferenciam o método definido neste capítulo com o método utilizado por Dougherty.

- inserimos a estimação do operador dentro do contexto de aprendizado PAC
- as aplicações não são restritas a problemas de restauração. Consideramos todos os operadores i.t. e localmente definidos
- utilizamos o conceito de função de perda, para caracterizar o erro de um operador (Dougherty utiliza somente o MAE)
- utilizamos o algoritmo ISI para calcular a base, que é muito melhor que o algoritmo QM para diversos casos reais.

No próximo capítulo descreveremos a implementação deste sistema. Todas as partes deste sistema estão baseadas em resultados teóricos formais e sua formulação é inédita.

Capítulo 8

Implementações

O sistema descrito no capítulo 7 foi implementado sobre um sistema denominado KHOROS, no ambiente UNIX. O objetivo deste capítulo é descrever as implementações realizadas e as especificações de uso deste sistema, além de sugerir melhorias de implementação para um desempenho mais eficiente do sistema.

Inicialmente apresentaremos uma descrição suscinta do sistema KHOROS e da "toolbox mmach". Em seguida descreveremos a implementação que corresponde à estrutura básica do sistema, conforme descrito no capítulo anterior, e descreveremos também estruturas que exploram o refinamento dos resultados através da composição de operadores.

8.1 Plataforma de Trabalho KHOROS

O sistema KHOROS ([RASW90]) é um software desenvolvido por uma equipe do "Department of Electrical and Computer Engineering" da "University of New Mexico", e que proporciona um ambiente para o desenvolvimento de programas e aplicações relacionados com o processamento de imagens.

Este sistema possui uma interface para o desenvolvimento de novos programas, o qual gerencia a criação dos programas fontes, sua documentação e geração dos programas executáveis. Além disso, possui também uma interface gráfica de alto nível, chamado CANTATA, que oferece um ambiente para programação visual.

Um conjunto de programas desenvolvidos neste ambiente pode ser agrupado e organizado como uma "toolbox" (caixa de ferramentas) e pode ser facilmente integrado ao sistema, de acordo com a necessidade do usuário. Os programas da toolbox podem ser acessados no CANTATA por meio de menus de seleção e ligados entre si de forma que a saída de um deles seja a entrada de outro. Estes programas são representados graficamente por um ícone (uma pequena figura quadriculada) e o fluxo de dados é representado por arestas orientadas ligando estes ícones (veja figura 8.1). Estes ícones ligados convenientemente representam um programa típico no cantata, os quais são denominados **workspaces**.

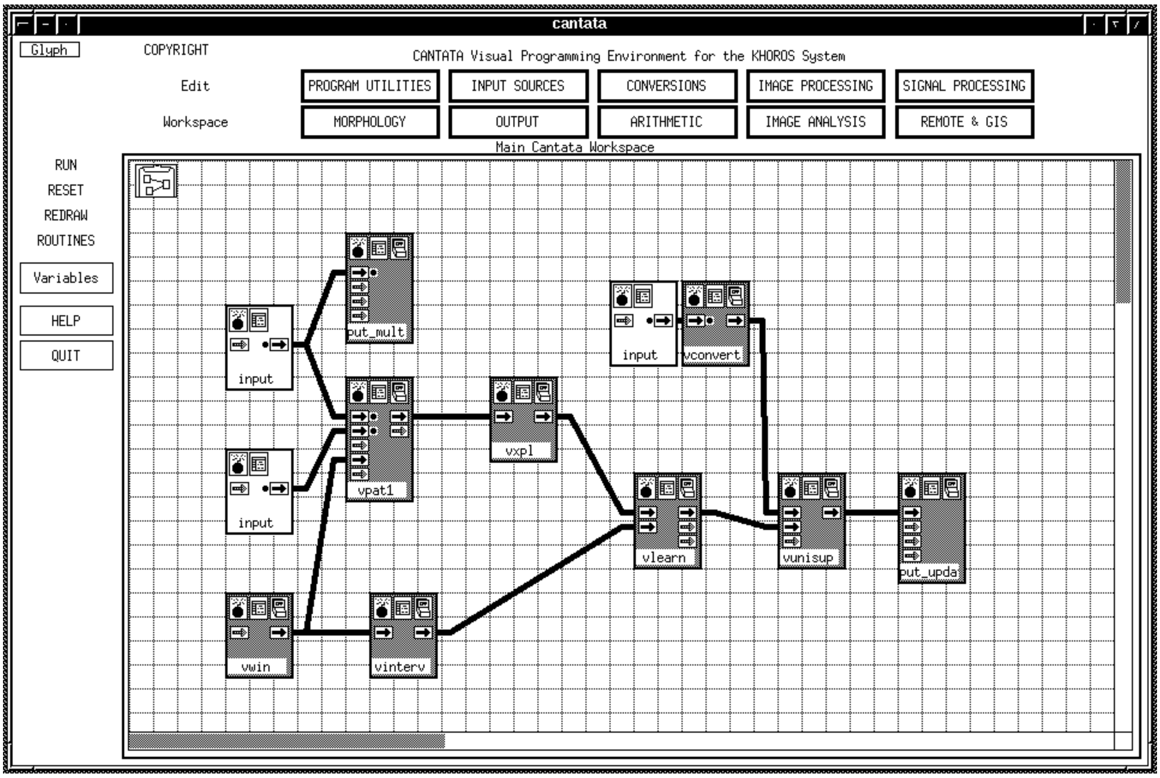


Figura 8.1: Um workspace no Cantata.

8.1.1 A Toolbox MMach

Existe um conjunto de programas, denominado MMach, que inclui operadores elementares e mais uma série de outros operadores úteis da *MM* (ver apêndice A em [BB94]). A MMach está implementada como uma toolbox (denominada, também, toolbox MMach) no ambiente KHOROS ([BBL94]).

Utilizaremos elementos da toolbox MMach para implementar os operadores aprendidos pelo sistema descrito no capítulo 7.

8.2 Softwares Implementados

Para implementar o sistema descrito no capítulo anterior, criamos um conjunto de programas escritos em linguagem de programação C, no ambiente KHOROS. Estes programas foram instalados como uma "toolbox" do CANTATA, denominada "PAC" e são divididos em três grupos : módulo de treinamento, módulo de aplicação e módulo de ferramentas auxiliares.

8.2.1 Módulo de Treinamento

O módulo de treinamento engloba rotinas necessárias para implementar as três partes do sistema.

1. **vwin** : especifica o tamanho e formato da janela W.

    - **Entrada** :
        - imagem binária especificando formato da janela, ou dimensão da janela no caso de janela retangular
    - **Saída** :
        - especificação da janela

2. **vpat** : realiza a coleta de exemplos de treinamento.

    - **Entrada** :
        - especificação de uma janela ou uma tabela de exemplos
        - imagens de treinamento (X, Y)
        - máscara para coleta de exemplos M (opcional)
    - **Saída** :
        - tabela de exemplos
        - relatório sobre exemplos coletados (opcional)

    As imagens de treinamento devem ser de mesmo tamanho para que, ao se transladar a janela W sobre X, seja possível associar um valor Y_z para cada configuração $X \cap W + z$. Caso seja especificado uma tabela de exemplos na entrada, então os novos exemplos coletados serão acrescentados a ela.

A máscara, que deve ser também uma imagem com as mesmas dimensões de X e Y, indica para quais $z \in E$ a correspondente configuração deve ser obtida (isto é, $X \cap W + z \in \mathcal{C}(X_W) \iff M_z = 1$). A especificação é útil, por exemplo, quando sabemos que o operador ψ a ser aprendido é um operador anti-extensivo, isto é, $\psi(X) \subseteq X$. Neste caso, $z \notin X \Longrightarrow z \notin \psi(X)$, e portanto somente as configurações que contém a origem (isto é, aqueles tais que $X_z = 1$) precisam ser analisadas. É possível coletar somente as configurações que satisfazem esta propriedade utilizando uma máscara $M = X$.

$T = \emptyset$ ou tabela de entrada ; (Tabela de exemplos)
Para cada $z \in E$ (onde E é o domínio da imagens)
    se $M_z = 1$ então
        calcule $x = X \cap W + z$ e Y_z ;
        se x está em T então
            se $Y_z = 1$ então $f_1(x) = f_1(x) + 1$;
            senão $f_0(x) = f_0(x) + 1$;
        senão
            insira x em T ;
            se $Y_z = 1$ então $f_1(x) = 1$;
            senão $f_0(x) = 1$;
Devolva T;

A tabela de exemplos foi implementada utilizando-se a estrutura de árvore balanceada ([Knu73, Amm92]) para tornar as buscas eficientes, uma vez que para cada configuração coletada, deve-se verificar se ela já está contida na tabela.

3. vnpat : realiza a coleta de exemplos de treinamento a partir de múltiplos pares de imagens.

- **Entrada** :
  - especificação de uma janela
  - conjunto de imagens de treinamento $\{(X_i, Y_i)\}$
  - conjunto de máscaras para coleta de exemplos $\{M_i\}$ (opcional)
- **Saída** :
  - tabela de exemplos
  - relatório sobre exemplos coletados (opcional)
  - histograma subtrativo (opcional)

Valem as mesmas observações feitas em relação a *vpat*. Aqui a diferença em relação ao *vpat* é a possibilidade de coletar exemplos a partir de vários pares de imagens de treinamento.

A análise da distribuição de configurações nas imagens de treinamento pode ser útil para definir a quantidade de imagens a serem utilizadas no treinamento. O "histograma subtrativo" é uma saída opcional que gera um conjunto de dados cujo gráfico correspondente pode ser interpretado da seguinte forma : a abscissa indica o número de imagens das quais foram coletadas as configurações. A ordenada correspondente a i é a quantidade de novas configurações distintas observadas em (X_i, Y_i) em relação às configurações distintas observadas em $\{(X_1, Y_1), ..., (X_{i-1}, Y_{i-1})\}$.

4. **vinterv** : define o(s) intervalo(s) de inicialização do algoritmo de aprendizado ISI.

- **Entrada** :
  - tipo de intervalo desejado : $[\emptyset, W]$ ou $[\{o\}, W]$
  - especificação de janela
- **Saída** :
  - especificação de intervalo

O intervalo $[\emptyset, W]$ corresponde aos operadores i.t. localmente definidos por W gerais, e o intervalo $[\{o\}, W]$ corresponde aos operadores i.t. localmente definidos por W e anti-extensivos (pois todos os intervalo que caracterizam um operador i.t. anti-extensivo estão contidos em $[\{o\}, W]$).

5. **vxpl** : gera uma tabela de mintermos a partir de uma tabela de exemplos.

- **Entrada** :
  - tabela de exemplos
  - critério de seleção $(0 \leq \alpha_1 \leq 1)$
  - critério de decisão $(0 \leq \alpha_2 \leq 1$, ação)
- **Saída** :
  - tabela com mintermos

O critério de seleção pode ser usado para rejeitar configurações pouco freqüentes. Se $f_0(x) + f_1(x) \leq \alpha_1$ então a configuração x não será inserida na tabela de mintermos.

O critério de decisão é utilizado para decidir se uma dada configuração é positiva ou negativa.

- se $f_0 > f_1$ e $\frac{f_0(x)}{f_0(x)+f_1(x)} > \alpha_2$ então $h(x) = 0$;

- se $f_1 > f_0$ e $\frac{f_1(x)}{f_0(x)+f_1(x)} \geq \alpha_2$ então $h(x) = 1$;

- caso contrário $\begin{cases} h(x) = 1 & \text{se ação = positivo} \\ h(x) = 0 & \text{se ação = negativo} \\ \text{rejeita } x & \text{caso contrário} \end{cases}$

6. **vlearn** : realiza a minimização a partir dos mintermos, utilizando o algoritmo ISI.

- **Entrada** :
  - tabela de mintermos
  - tipo do algoritmo (ISI-0, ISI-1, ISI-2, ISI-3)
- **Saída** :
  - base ótima reduxida (ou sub-ótima) - implicantes primos essenciais
  - base completa (opcional para ISI-0 e ISI-1) - implicantes primos
  - relatório (opcional)

8.2.2 Módulo de Aplicação

O módulo de aplicação corresponde à implementação do operador gerado pelo sistema, utilizando elementos da toolbox Mmach.

1. **vunisup** : realiza a aplicação do resultado do aprendizado (união de sup-geradoras). Este programa utiliza algumas rotinas da "toolbox" MMach.

- **Entrada** :
  - base de um operador ($\mathcal{B}$)
  - imagem binária (X)
- **Saída** :
  - resultado da aplicação do operador sobre a imagem ($\psi(X)$)

Calcula $\psi(X) = \bigcup\{\lambda^W_{(A,B)}(X) : [A,B] \in \mathcal{B}\}$.

8.2.3 Módulo de Ferramentas Auxiliares

Módulo de ferramentas auxiliares são programas que facilitam a análise de dados obtidas durante o processo de treinamento.

1. **vdspbas** : mostra os elementos de uma base.

   - **Entrada** :
     - base de um operador
     - quantidade de elementos para serem mostrados
   - **Saída** :
     - formato geométrico dos elementos da base

   Um elemento $A \in W$ é codificado e gravado em notação hexadecimal. A seguir mostramos a codificação dos elemento de uma base, onde W é um quadrado 3×3, e sua respectiva forma geométrica exibida por *vdspbas*.

```
BASE 9 4
WINSPEC     5     5     9
739c0
10   100
10   40
10   4
10   1
```

```
VISUALIZATION OF BASIS
 Total of base elements = 4

     0 0 0         1 1 1
 1)  0 1 0         1 1 1
     0 0 0         1 1 0

     0 0 0         1 1 1
 2)  0 1 0         1 1 1
     0 0 0         0 1 1

     0 0 0         1 1 0
 3)  0 1 0         1 1 1
     0 0 0         1 1 1
```

```
    0 0 0      0 1 1
4)  0 1 0      1 1 1
    0 0 0      1 1 1
```

2. **vdspcfg** : mostra os exemplos de treinamento coletados por *vpat1* ou *vnpat*.
   - **Entrada** :
     - tabela de exemplos
     - quantidade de elementos para serem mostrados
   - **Saída** :
     - formato geométrico dos elementos da tabela de exemplos
3. **vdspxpl** : mostra a configuração associada aos mintermos gerados por *vxpl*.
   - **Entrada** :
     - tabela de mintermos
     - quantidade de mintermos para serem mostrados
   - **Saída** :
     - formato geométrico dos mintermos
4. **vmerge** : realiza a união de duas tabelas de exemplos.
   - **Entrada** :
     - duas tabelas de exemplos
   - **Saída** :
     - união das duas tabelas de exemplos
5. **vhist** : cria histograma associado a uma tabela de exemplos.
   - **Entrada** :
     - tabela de exemplos
   - **Saída** :
     - histograma correspondente à tabela de exemplos.

   Este programa ordena os exemplos da tabela de exemplos por freqüência, e gera um conjunto de dados que representam o gráfico da distribuição de exemplos. Estes dados podem ser visualizados através de uma das opções do CANTATA (ver figura 8.2).

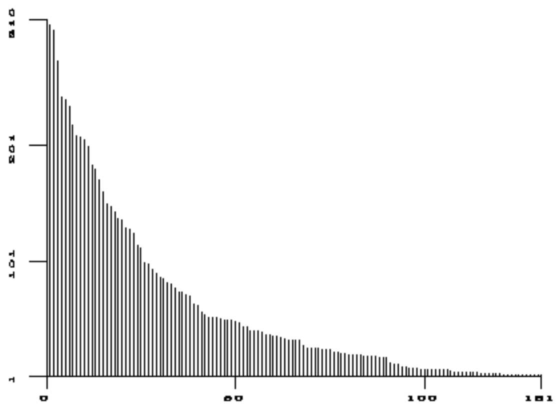


Figura 8.2: Histograma da distribuição de exemplos.

8.3 Estrutura Básica

Para implementar a estrutura básica do sistema, os programas listados acima devem ser convenientemente "ligados", conforme ilustra a figura 8.3. As imagens de treinamento (X, Y) e a especificação de uma janela devem alimentar o programa *vpat1* (ou *vnpat*), que realiza a coleta de exemplos. O programa *vxpl* decide qual das configurações coletadas serão positivos e quais serão negativos de acordo com um critério de decisão especificado pelo usuário. O resultado do aprendizado, gerado pelo programa *vlearn*, é a base do operador estimado. Para aplicar este operador, este deve alimentar o programa *vunisup* juntamente com uma imagem Z que se deseja processar.

8.4 Estrutura Seqüencial

A implementação seqüencial da estrutura básica será denominada estrutura seqüencial. Do ponto de vista teórico, a estrutura seqüencial corresponde à composição de operadores e do ponto de vista prático corresponde a um processo sucessivo de refinamento do resultado obtido.

Sejam $I_1, I_2, ..., I_n$ conjuntos de índices disjuntos 2 a 2 e sejam $\{X_i : i \in I_j\}$, $j \in \{1, ..., n\}$ e $\{Y_i : i \in I_j\}$ coleções de realizações dos conjuntos aleatórios $\boldsymbol{X}$ e $\boldsymbol{Y}$ respectivamente, isto é, (X_i, Y_i) são imagens de treinamento. Suponha que as imagens das coleções correspondentes a I_1 são utilizados para aprender um operador ψ_1. Os pares do tipo $(\psi_1(X_i), Y_i)$, $i \in I_2$, podem ser utilizados para treinar um segundo operador ψ_2, que age como um filtro sobre $\psi_1(\cdot)$. Os pares $(\psi_2(\psi_1(X_i)), Y_i)$, $i \in I_3$, podem ser utilizados

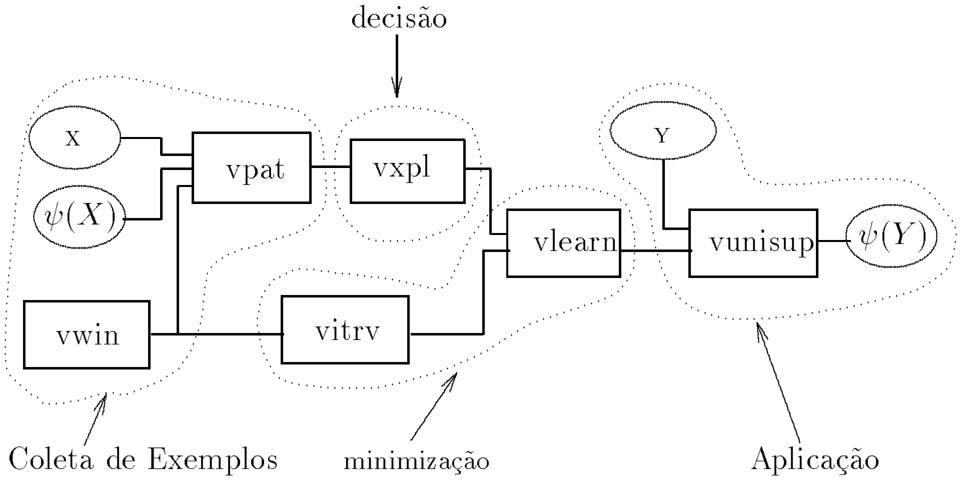


Figura 8.3: Estrutura básica.

para treinar ψ_3 e assim sucessivamente. Esta dinâmica determina a estrutura seqüencial do sistema e o número de operadores treinados determina o número de estágios da estrutura. O operador final ψ após n estágios é o operador obtido a partir da composição dos n operadores aprendidos, isto é, $\psi = \psi_n \cdots \psi_2\psi_1$.

Para implementar a estrutura seqüencial, não necessitamos de nenhum programa adicional, pois basta utilizarmos repetidas vezes a estrutura básica descrita na seção anterior.

Ilustramos na figura 8.4 a implementação de uma estrutura seqüencial. O número de estágios de um treinamento nesta estrutura não pode ser pré-estabelecido; ele depende de uma série de fatores como por exemplo a característica das imagens, o tamanho da janela do primeiro estágio, entre outros, conforme verificamos experimentalmente (capítulo 10).

Os operadores gerados pela estrutura básica do sistema possuem uma representação estritamente paralela (união de sup-geradoras), enquanto os gerados pela estrutura seqüencial possuem uma estrutura paralelo-seqüencial (composição de operadores, cada qual representados por uma união de sup-geradoras).

A utilização deste sistema envolve várias questões como a escolha da janela e critério de decisão, entre outras questões já mencionadas nos capítulos anteriores. Nos próximos dois capítulos apresentaremos vários exemplos de aplicação deste sistema, tentando inclusive abordar estas questões.

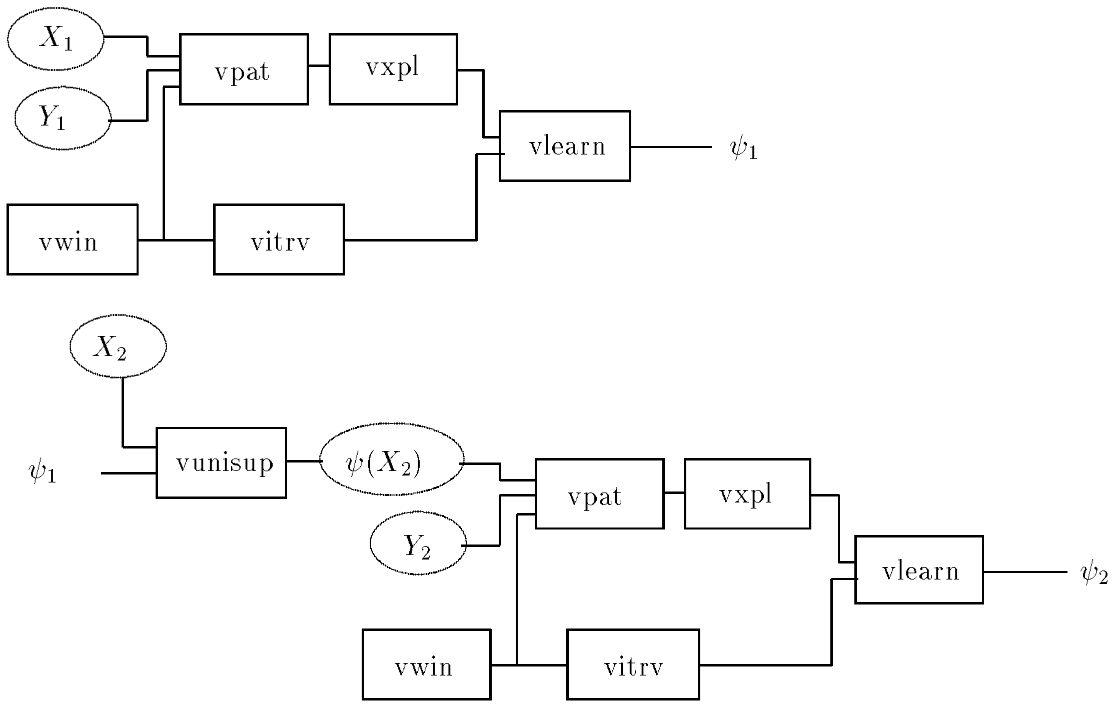


Figura 8.4: Estrutura seqüencial com 2 estágios.

Capítulo 9

Alguns Exemplos de Aplicação do Sistema

Neste capítulo apresentaremos uma série de exemplos de aplicações do sistema descrito no capítulo 7, cuja implementação foi realizada conforme descrição apresentada no capítulo 8.

Esta série de exemplos ilustra a qualidade de "abrangência", citada como uma das características desejáveis em um sistema para programação automática de MMach's, isto é, o sistema pode ser utilizado para resolver vários tipos de problemas no domínio das imagens binárias.

Em cada um dos exemplos, apresentaremos os experimentos realizados através de uma tabela que descreve o tamanho da janela, o número de exemplos coletados (m), o número de exemplos distintos na amostra coletada (*exemplos distintos*), o número de exemplos contraditórios distintos observados na amostra coletada (*exemplos contraditórios*), o número de exemplos negativos distintos (*exemplos negativos*), o número de exemplos positivos distintos (*exemplos positivos*), o algoritmo de aprendizado utilizado (*algoritmo*), o tempo gasto pelo algoritmo de aprendizado (*tempo*), o número máximo de intervalos simultâneos na memória durante a execução do algoritmo de aprendizado (*memória*) e o tamanho da base obtida (*base*). O tempo de processamento é descrito em termos de horas(h), minutos(m) e segundos(s).

9.1 Reconhecimento de Pontos Extremos

Neste exemplo, o objetivo consiste em identificar os pontos extremos de segmentos de reta, cuja expessura é de 1 pixel, sem levar em consideração a inclinação dos mesmos. Este tipo de operação pode ser útil, por exemplo, para identificar pontos de conexão em placas de circuitos elétricos, medir o comprimento de um objeto a partir do seu esqueleto, etc.

9.1.1 Pontos Extremos de Segmentos de Retas

As imagens de treinamento devem expressar a idéia da operação desejada. Neste sentido, elas podem ser bastante simples desde que contenham informações (exemplos) relevantes como ilustram as imagens da figura 9.1.

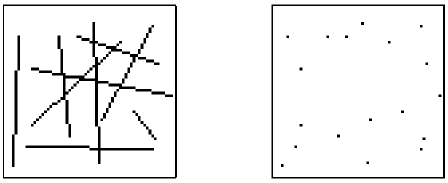


Figura 9.1: Imagens de treinamento relativos ao reconhecimento de pontos extremos de segmentos de reta - dimensão da imagem 64×64 pixels.

Utilizamos as imagens da figura 9.1 para realizar o treinamento, conforme descrição da tabela 9.1. Os elementos da base obtida, listados a seguir, foram aplicados sobre a

Parâmetro	valor ou medida
janela	3×3
m	$3,844$
exemplos distintos	94
exemplos contraditórios	0
exemplos negativos	88
exemplos positivos	6
algoritmo	ISI-3
tempo	0.6s
memória	16
base	4

Tabela 9.1: Experimento relativo à extração de bordas.

imagem da figura 9.2a e o resultado obtido é ilustrado na figura 9.2b.

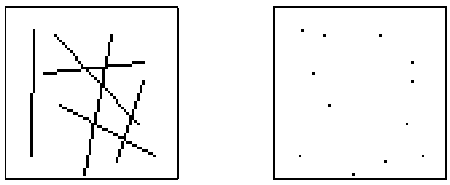
A B

```
0 0 0    0 0 0
0 1 0    1 1 0
0 0 0    0 1 0

0 0 0    0 1 0
0 1 0    0 1 1
0 0 0    0 0 0

0 0 0    0 0 0
0 1 0    0 1 0
0 0 0    1 0 1

0 0 0    1 0 1
0 1 0    0 1 0
0 0 0    0 0 0
```



a b

Figura 9.2: Resultado relativo à extração de pontos extremos - dimensão da imagem 64×64 pixels.

Observe que todos os pontos extremos, e somente os pontos extremos, foram extraídos. Este é um exemplo nos quais as amostras de treinamento são consistentes, e portanto ela recai no modelo PAC básico. A função de perda utilizada é o MAE.

9.2 Extração de Bordas

Em uma imagem binária, um ponto é considerado ponto de borda se possui pelo menos um ponto vizinho com valor distinto dele. O conjunto de pontos de borda que pertencem (estão contidos) nos objetos da imagem formam as bordas internas e as outras formam as bordas externas. A operação de extração de bordas pode ser útil para estabelecer o limite de cada objeto na imagem, classificar os objetos quanto à sua forma, etc.

Realizamos dois experimentos no que diz respeito a extração de bordas internas, com o intuito de ilustrar a importância de riqueza de informações nas imagens de treinamento.

9.2.1 Primeiro Experimento

Consideramos inicialmente o experimento realizado com as imagens de treinamento ilustradas na figura 9.3 :

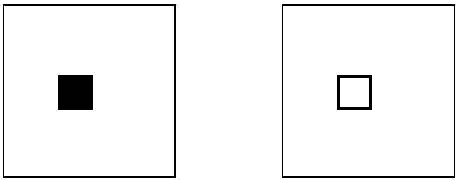


Figura 9.3: Imagens de treinamento para o primeiro experimento relativo à extração de bordas - dimensão da imagem 64×64 pixels.

A tabela 9.2 descreve o experimento realizado com as imagens de treinamento da figura 9.3.

Parâmetro	valor ou medida
janela	3×3
m	$3,844$
exemplos distintos	26
exemplos contraditórios	0
exemplos negativos	18
exemplos positivos	8
algoritmo	ISI-3
tempo	0.5s
memória	10
base	2

Tabela 9.2: Primeiro experimento relativo à extração de bordas.

Foi obtida uma base (base 1) de 2 elementos, listados a seguir :

```
      0 0 0          1 1 1
1)    0 1 0          1 1 1
```

```
      0 0 0        1 1 0

      0 0 0        0 1 1
2)    0 1 0        1 1 1
      0 0 0        1 1 1
```

9.2.2 Segundo Experimento

Em um segundo experimento de extração de bordas, utilizamos um outro par de imagens de treinamento, ilustrados na figura 9.4. Note que há um maior número (variação) de exemplos em relação ao primeiro experimento.

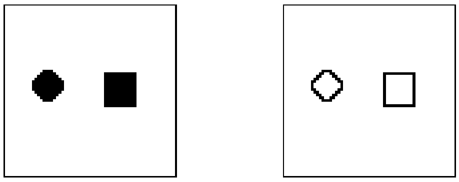


Figura 9.4: Imagens de treinamento para o segundo experimeto relativo à extração de bordas - dimensão da imagem 64×64 pixels.

A tabela 9.3 descreve o segundo experimento realizado.

Parâmetro	valor ou medida
janela	3×3
m	$3,844$
exemplos distintos	46
exemplos contraditórios	0
exemplos negativos	22
exemplos positivos	24
3algoritmo	ISI-3
tempo	0.5s
memória	10
base	4

Tabela 9.3: Segundo experimento relativo à extração de bordas.

Foi obtida uma base (base 2) com 4 elementos, listados a seguir :

```
        0 0 0        1 1 1
1)      0 1 0        1 1 1
        0 0 0        1 1 0

        0 0 0        1 1 1
2)      0 1 0        1 1 1
        0 0 0        0 1 1

        0 0 0        1 1 0
3)      0 1 0        1 1 1
        0 0 0        1 1 1

        0 0 0        0 1 1
4)      0 1 0        1 1 1
        0 0 0        1 1 1
```

Aplicamos as bases 1 e 2 sobre a imagem da figura 9.5a e obtivemos, respectivamente, as imagens ilustradas nas figuras 9.5b e 9.5c.

Observe que a base do segundo experimento fornece um resultado perfeito (figura 9.5c), enquanto que a base do primeiro experimento apresenta pequenas falhas (não reconheceu todos os pontos de borda - figura 9.5b). Esta diferença nos resultados é devido à expressividade dos exemplos contidos nas imagens de treinamento, ou seja, quanto mais expressivos (no sentido de representar bem a operação desejada) forem as imagens de treinamento, melhor é o resultado final.

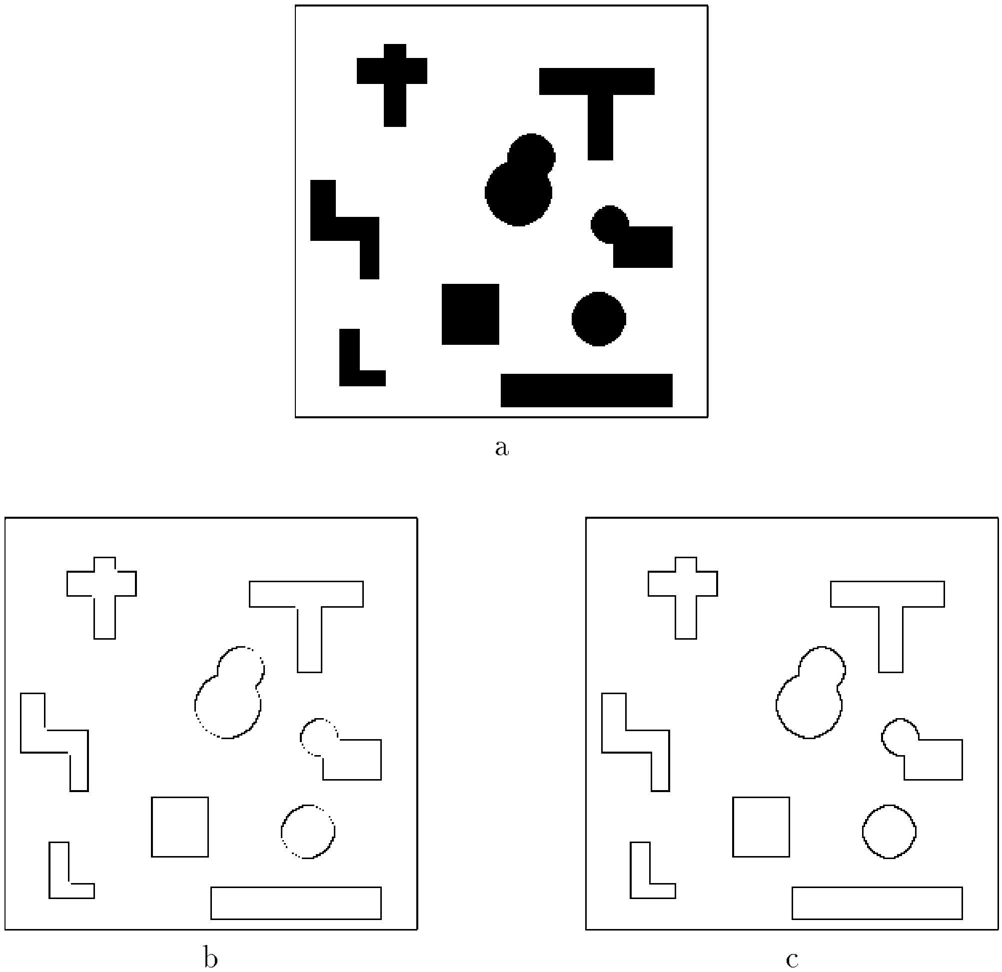


Figura 9.5: Resultado relativo à extração de bordas - dimensão da imagem 256 $\times$ 256 pixels.

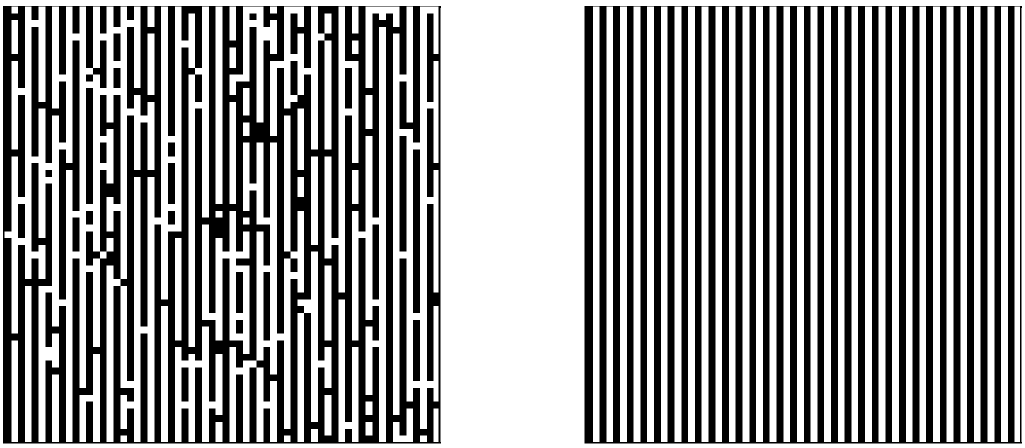
9.3 Restauração

A restauração de imagens corrompidas por ruídos é um problema amplamente pesquisado ([GW92, Hal79]). Dougherty, por exemplo, dedicou especial atenção ao projeto de filtros ótimos (veja capítulo 4). Realizamos alguns experimentos para restauração de imagens com ruídos puntuais de distribuição uniforme. Para podermos aplicar o método de aprendizado proposto sobre problemas de restauração, é necessário que a imagem a ser filtrada possua uma distribuição homogênea. A condição de estacionaridade é fundamental para conseguirmos bons resultados utilizando operadores invariantes por translação.

A função de perda que deve ser utilizada nos problemas de restauração é o MAE.

9.3.1 Filtragem de Ruídos sobre Listras

O primeiro experimento é semelhante ao apresentado por Dougherty em [Dou92a]. Criamos uma imagem listrada (figura 9.6b) e adicionamos ruídos aditivos e subtrativos puntuais de distribuição uniforme, de densidade 5%, conforme ilustra a figura 9.6.



a b

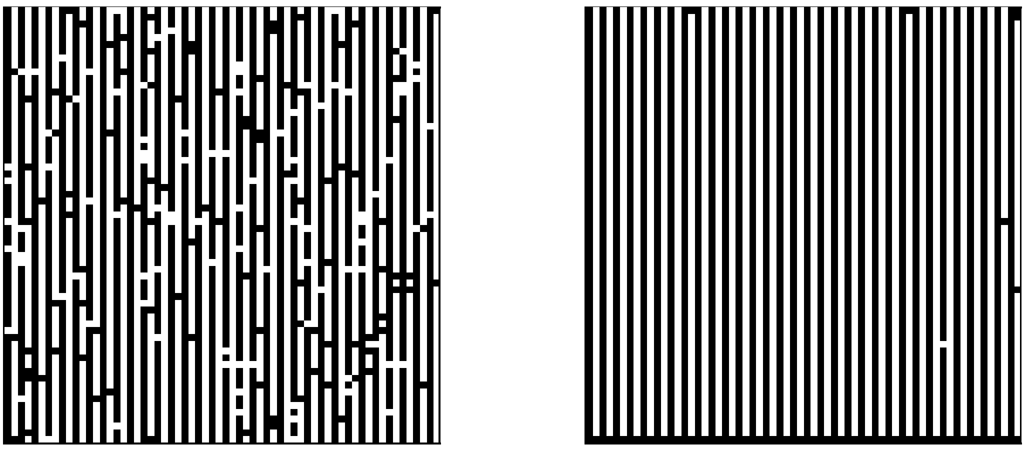
Figura 9.6: Imagens de treinamento relativo à filtragem de ruído sobre listras - dimensão da imagem 64×64 pixels.

O treinamento realizado sobre estas imagens é descrito na tabela 9.4.

A base obtida, com 5 elementos, foi aplicada sobre a imagem da figura 9.7a, que consiste da mesma imagem usada no treinamento, a menos dos ruídos (de mesma distribuição) que foram gerados novamente. O resultado obtido está ilustrado na figura 9.7b. Desconsiderando os pontos da primeira linha e coluna e última linha e coluna (que sofrem efeitos de borda) o operador errou em apenas dois pixels.

Parâmetro	valor ou medida
janela	3×3
m	$3,844$
exemplos distintos	119
exemplos conflitantes	1
exemplos negativos	56
exemplos positivos	63
algoritmo	ISI-3
tempo	0.7s
memória	12
base	5

Tabela 9.4: Experimento relativo à filtragem de ruído - dimensão da imagem 64×64 pixels.



a b

Figura 9.7: Resultado da filtragem de ruído sobre listras.

9.3.2 Filtragem de Ruídos sobre Caracteres

Neste experimento, consideramos um trecho de texto capturado diretamente do monitor de vídeo. Sobre a imagem assim obtida, adicionamos ruídos aditivos e subtrativos puntuais, de distribuição uniforme e densidade 5%, conforme ilustra a figura 9.8.

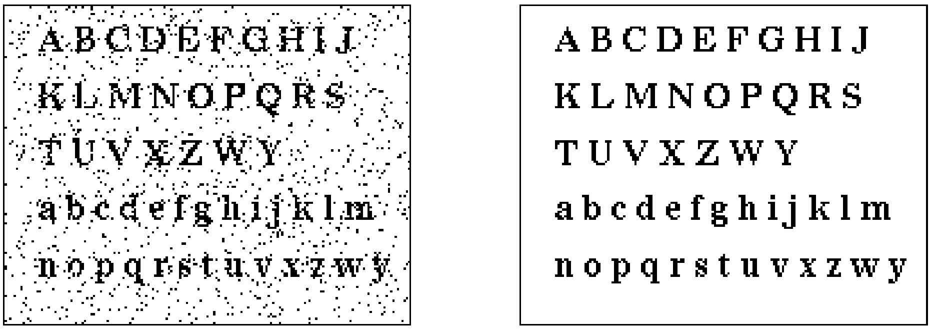


Figura 9.8: Imagens de treinamento relativo à filtragem de ruído sobre letras - dimensão da imagem 140×179 pixels.

Realizamos dois experimentos sobre dois pares de imagens de treinamento, variando apenas o tamanho da janela, conforme descrevemos nas tabelas 9.5 e 9.6.

Parâmetro	valor ou medida
janela	3×3
m	$48,852$
exemplos distintos	482
exemplos contraditórios	174
exemplos negativos	247
exemplos positivos	222
algoritmo	ISI-3
tempo	0.95s
memória	48
base	28

Tabela 9.5: Primeiro experimento relativo à filtragem de ruído sobre letras.

O primeiro treinamento resultou em uma base de 28 elementos e o segundo em uma de 245 elementos. Os resultados da aplicação destas bases sobre a imagem da figura 9.9a são ilustrados respectivamente nas figuras 9.9b e 9.9c. O resultado referente à janela

Parâmetro	valor ou medida
janela	5×5
m	$47,600$
exemplos distintos	$12,847$
exemplos contraditórios	26
exemplos negativos	$9,490$
exemplos positivos	$3,341$
algoritmo	ISI-3
tempo	$10m49s$
memória	$1,080$
base	245

Tabela 9.6: Segundo experimento relativo à filtragem de ruído sobre letras.

3×3 parece, aparentemente, melhor do que o referente à janela 5×5, o que contradiz o resultado esperado. Porém, se analisarmos a quantidade de exemplos distintos nos dois casos, podemos constatar que : no caso da janela 3×3 observamos 482 exemplos distintos em $2^9 = 512$ possíveis, ou seja, mais de 94%; no caso da janela 5×5, observamos $12,847$ em $2^{25} = 33,554,432$ possíveis, ou seja, menos de 1%. Portanto, a probabilidade de uma configuração não observada no treinamento ser observada na aplicação é muito maior no caso da janela 5×5.

Aplicamos estas duas bases sobre a imagem usada no treinamento e os resultados obtidos são mostrados na figura 9.10. Em situações nos quais os exemplos não são contraditórios o resultado assim obtido deve ser perfeito, uma vez que o operador aprendido deve ser consistente com os exemplos de treinamento. Entretanto, quando há contradições nos exemplos, o operador é consistente com a decisão tomada. Os resultados relativos às janela 3×3 e 5×5 são apresentados, respectivamente, nas figuras 9.10a e 9.10b. O resultado referente à janela 5×5 é melhor que o referente à janela 3×3, uma vez que a amostra referente à primeira contém menos exemplos contraditórios.

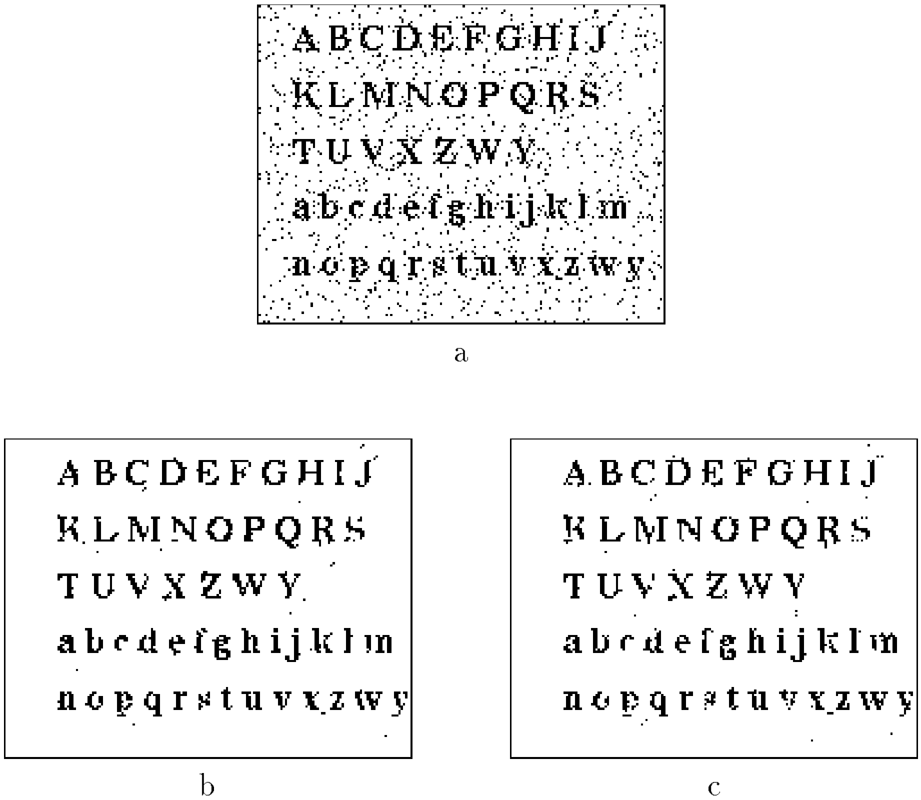


Figura 9.9: Resultado da filtragem de ruído sobre letras - dimensão da imagem 140×179 pixels.

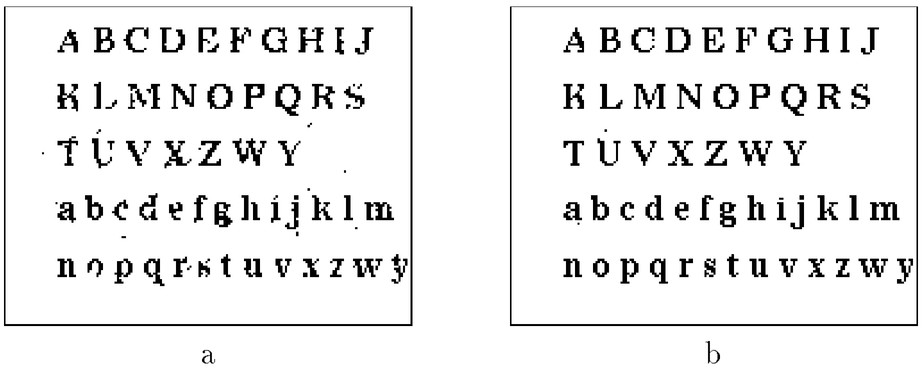


Figura 9.10: Aplicação sobre imagem de treinamento - dimensão da imagem 140×179 pixels.

9.4 Restauração e Extração

Nesta seção apresentaremos exemplos de aplicação combinada, isto é, a aplicação do sistema para resolver problemas que consistem da combinação de dois problemas básicos.

9.4.1 Extração de Bordas de Imagens com Ruído

Suponha que desejamos extrair a borda (contorno) das figuras de uma determinada imagem (este problema foi ilustrado na seção 9.2). Vamos supor também que estas imagens foram corrompidas por algum tipo de ruído. Nesta situação, o problema combina dois casos básicos : extração de bordas e restauração.

Considere os dois pares de imagens de treinamento, ilustrados na figura 9.11. Os ruídos, aditivos e subtrativos, são de distribuição uniforme e de densidade 5%.

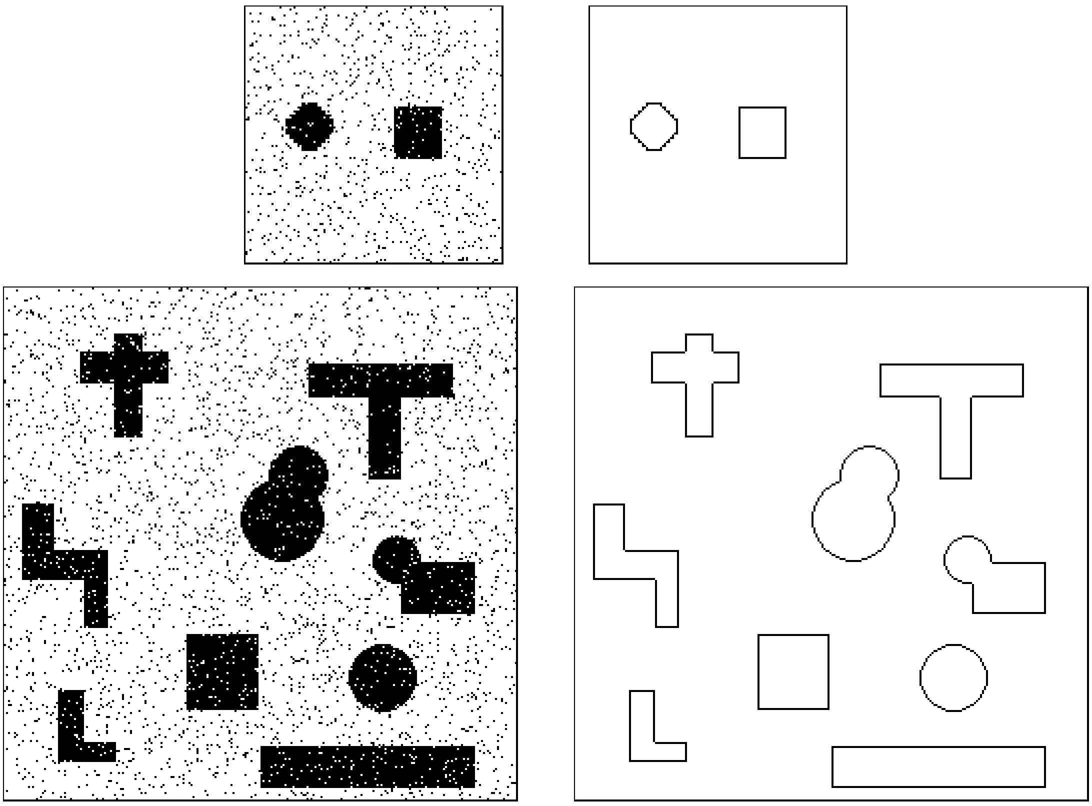


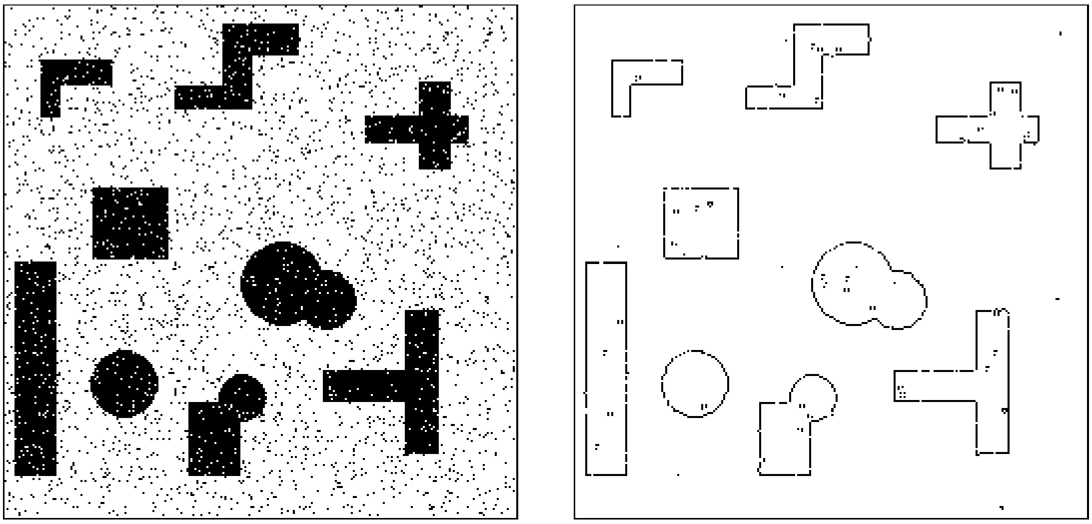
Figura 9.11: Imagens de treinamento relativo à restauração e extração de bordas - dimensão da imagem 128×128 e 256×256 pixels, respectivamente.

A tabela 9.7 descreve o experimento realizado a partir das imagens de treinamento ilustrados na figura 9.11.

Parâmetro	valor ou medida
janela	3×3
m	$80,392$
exemplos distintos	378
exemplos contraditórios	91
exemplos negativos	293
exemplos positivos	85
algoritmo	ISI-3
tempo	$1.5s$
memória	289
base	29

Tabela 9.7: Experimento relativo à filtragem e extração de bordas.

Na figura 9.12 mostramos o teste realizado com a base obtida, sobre uma outra imagem com as mesmas características da imagem utilizada no treinamento.



a b

Figura 9.12: Resultado da filtragem e extração de bordas - dimensão da imagem 256×256 pixels.

9.5 Reconhecimento de Textura

Uma textura é caracterizada pela repetição de um determinado padrão. Por exemplo, uma área preenchida com linhas horizontais paralelas e equidistantes entre si, ou uma área preenchida com linhas horizontais e verticais, ou áreas preenchidas com bolinhas, listras, pontilhados ou pequenos padrões geométricos, caracterizam várias texturas. Estas texturas são importantes, por exemplo, para demarcar diferentes regiões (baseadas em divisão política, tipo de solo, condições sócio-econômicas, entre outros) em uma figura de um mapa, para distinguir diferentes elementos em um gráfico, etc.

Em processamento de imagens, a análise de texturas permite reconhecer regiões de interesse tais como áreas de cultivo de determinado produto agrícola em imagens obtidas por satélites.

Neste exemplo ilustramos a aplicação do sistema para reconhecimento de texturas. Utilizamos uma parte de uma imagem (figura 9.13) para realizar o treinamento do operador e em seguida aplicamos o operador aprendido sobre a imagem inteira (figura 9.14). A tabela 9.8 descreve o treinamento realizado a partir das imagens ilustradas na figura 9.13. O resultado obtido está ilustrado na figura 9.15.

Parâmetro	valor ou medida
janela	5×5
m	$90,216$
exemplos distintos	$5,098$
exemplos contraditórios	19
exemplos negativos	$5,024$
exemplos positivos	74
algoritmo	ISI-3
tempo	0.79s
memória	64
base	12

Tabela 9.8: Experimento relativo ao reconheciemnto de texturas.

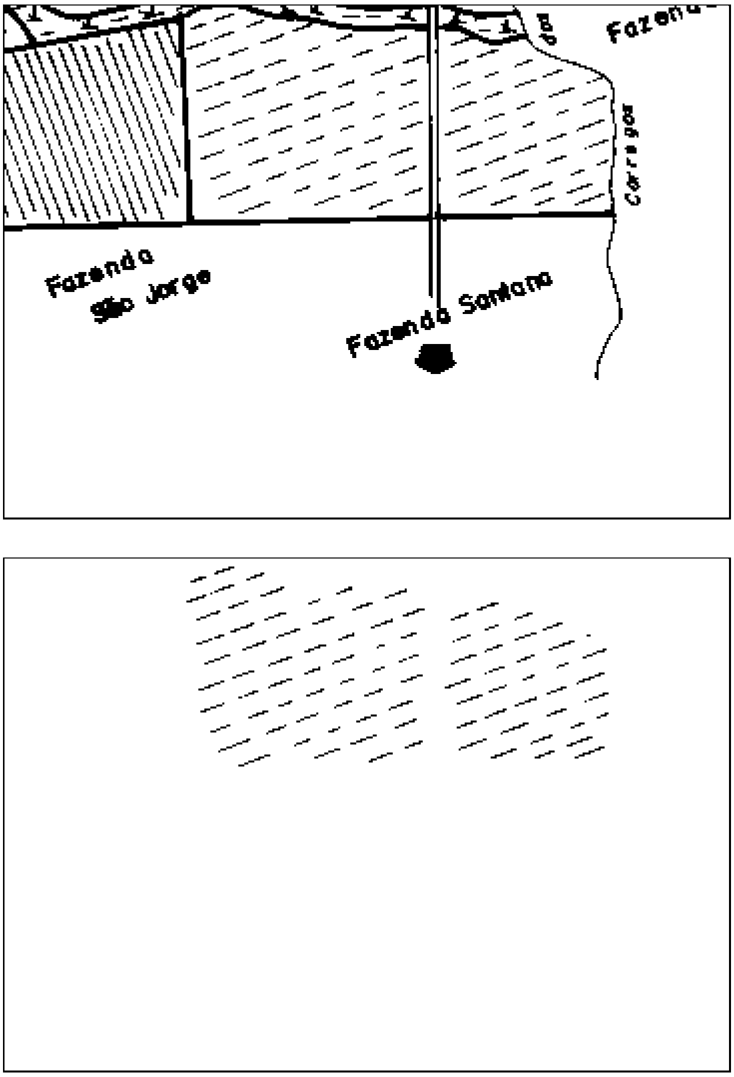


Figura 9.13: Imagens de treinamento para reconhecimento de textura - dimensão da imagem 256×362 pixels.

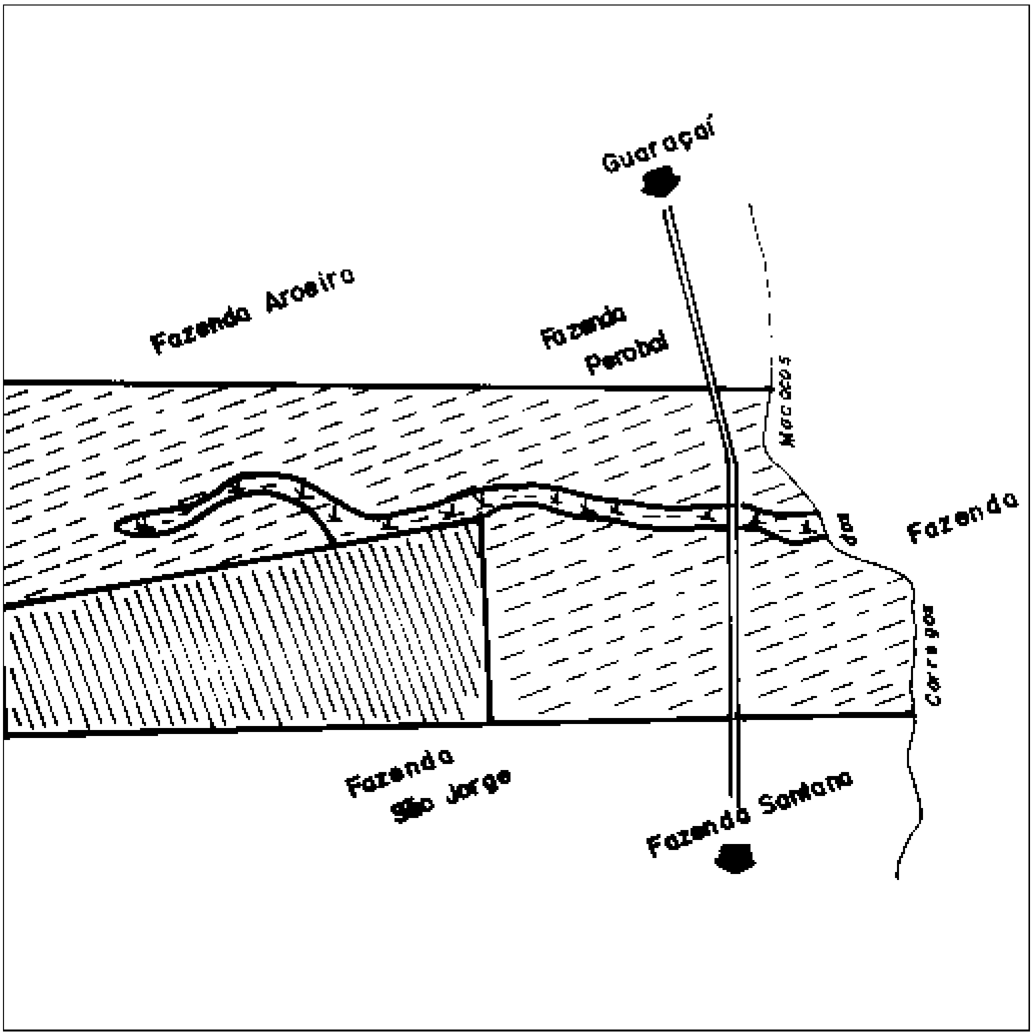


Figura 9.14: Imagem para testar operador de reconhecimento de textura - dimensão da imagem 512 $\times$ 512 pixels.

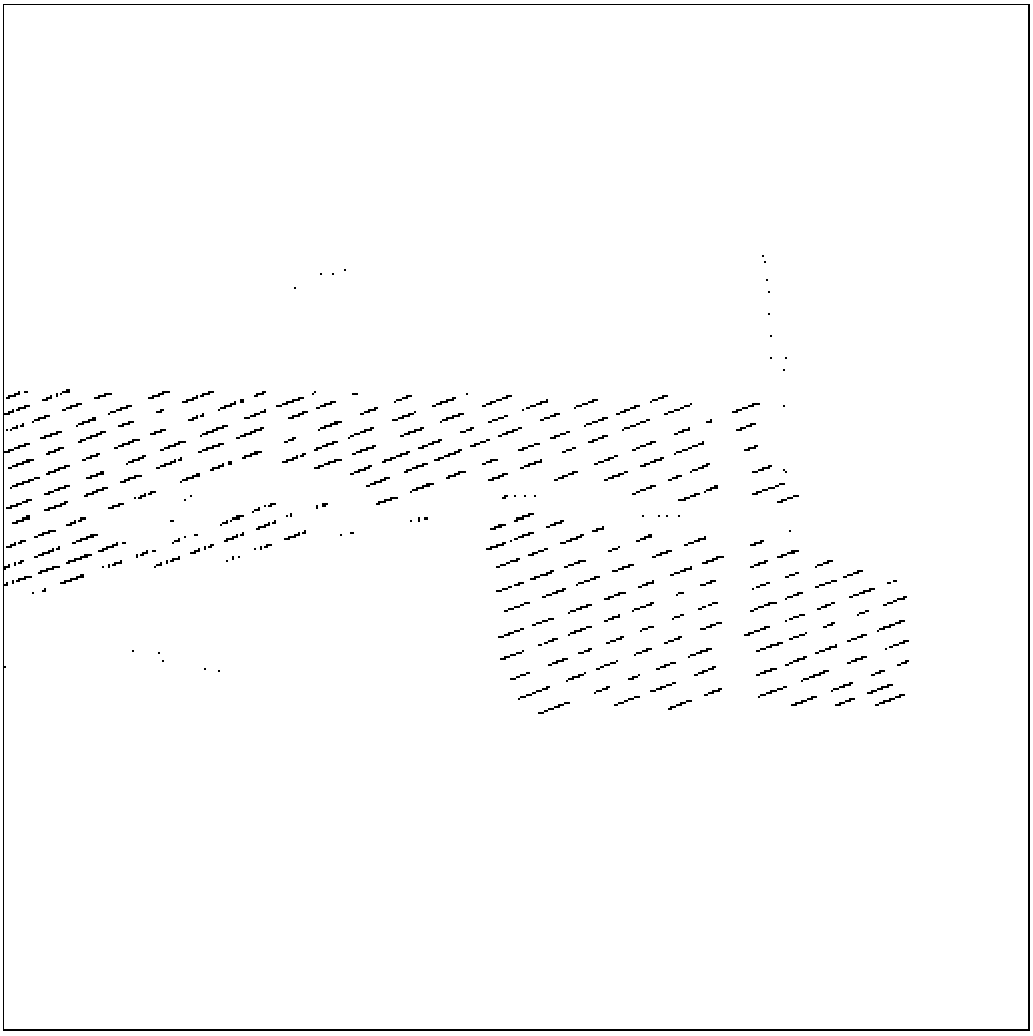


Figura 9.15: Resultado relativo ao reconhecimento de textura - dimensão da imagem 512×512 pixels.

9.6 Localização de Código de Barras

Em algumas aplicações de processamento de imagens, o primeiro processamento consiste em isolar a área de interesse numa imagem. Por exemplo, se a aplicação é o reconhecimento automático de CEP nos envelopes, é desejável que o reconhecedor seja capaz de localizar a região no envelope que contém o CEP, para, a partir dele, passar para o processo de reconhecimento do CEP propriamente dito. O mesmo vale para problemas de autenticação de assinaturas de um documento, leitura de código de barras, etc.

Estamos interessados em localizar uma subregião da imagem que corresponde à área que contém um código de barras. Dependendo do tipo de documento, para que as características que distinguem o código de barras de outros elementos no documento não seja perdido, muitas vezes é necessário adquirir a imagem em alta resolução (o que aumenta o tamanho da imagem).

A figura 9.16 ilustra uma imagem correspondente a um documento contendo um código de barras. A dimensão original, $1,772 \times 799$, pixels foi reduzida para 768×370 para poder ser ilustrada neste texto.

Para aplicar o processo de aprendizado neste caso, o tamanho da janela deve ser grande, o que pode não ser viável na prática. Por outro lado, reduzir a escala da imagem não representa uma boa alternativa pois este procedimento deforma as figuras presentes na imagem.

A abordagem encontrada para este caso consiste em realizarmos um pré-processamento na escala original da imagem (resolução alta). Com a aplicação de uma seqüência de operadores denominados fechamentos (isto é, dilatação seguida de erosão) sobre a imagem original, foi possível gerar imagens como a ilustrada do lado esquerdo na figura 9.17. Observe que nessa imagem, muitos elementos sofreram deformações, inclusive o código de barras. Porém a informação principal, a área que contém o código de barras, é preservada.

A tabela 9.9 descreve o experimento realizado com as imagens de treinamento ilustrados na figura 9.17 e a figura 9.18 mostra, respectivamente, a imagem no qual a base foi aplicada, e o resultado obtido.

Parâmetro	valor ou medida
janela	11×11
m	$20,764$
exemplos distintos	$13,067$
exemplos contraditórios	0
exemplos negativos	$11,659$
exemplos positivos	$1,408$
algoritmo	ISI-3
tempo	1m28s
base	23

Tabela 9.9: Experimento relativo à localização de código de barras.

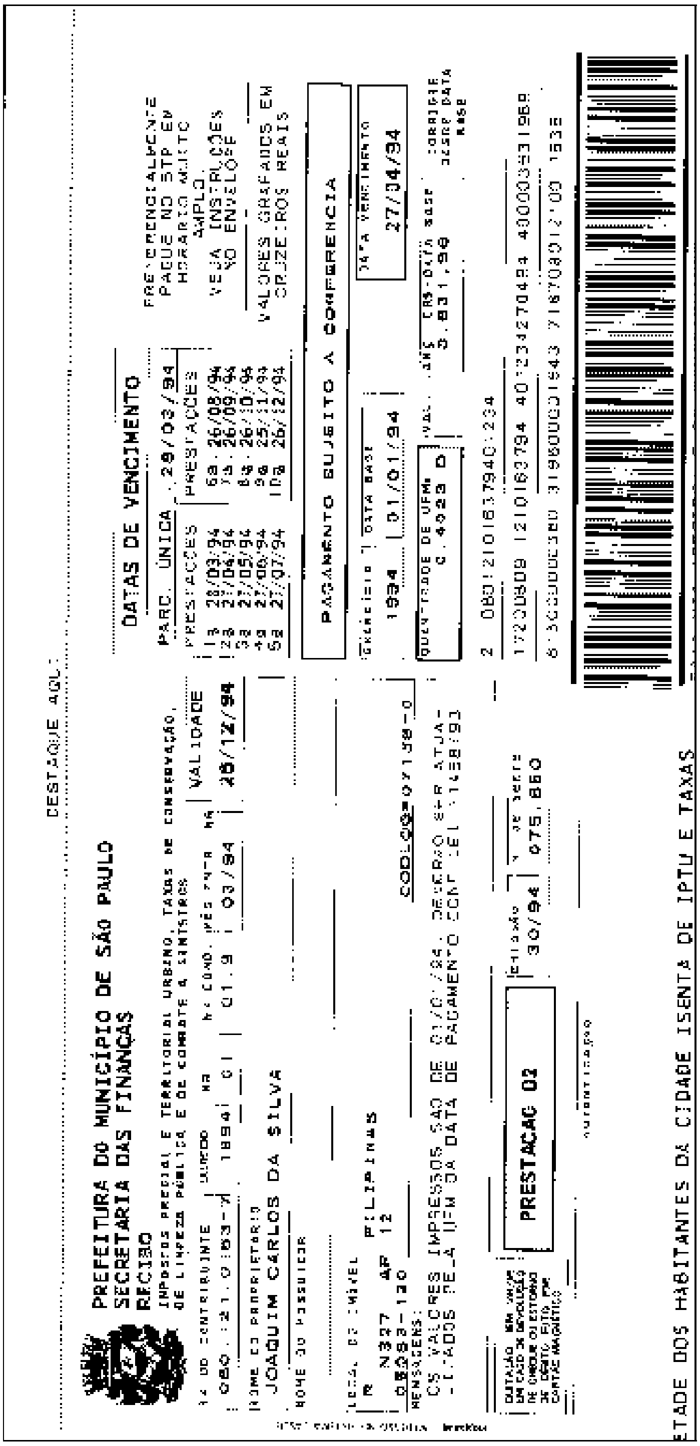


Figura 9.16: Imagem utilizada para localização de código de barras - dimensão 768 × 370 pixels.

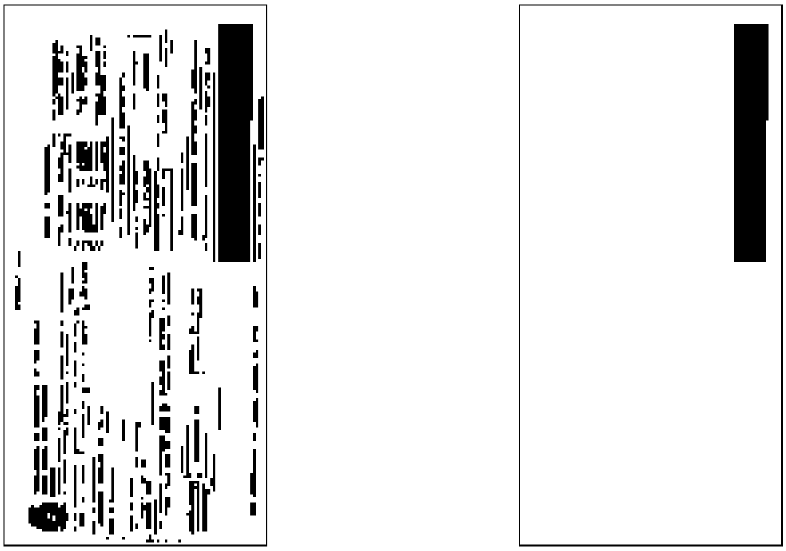
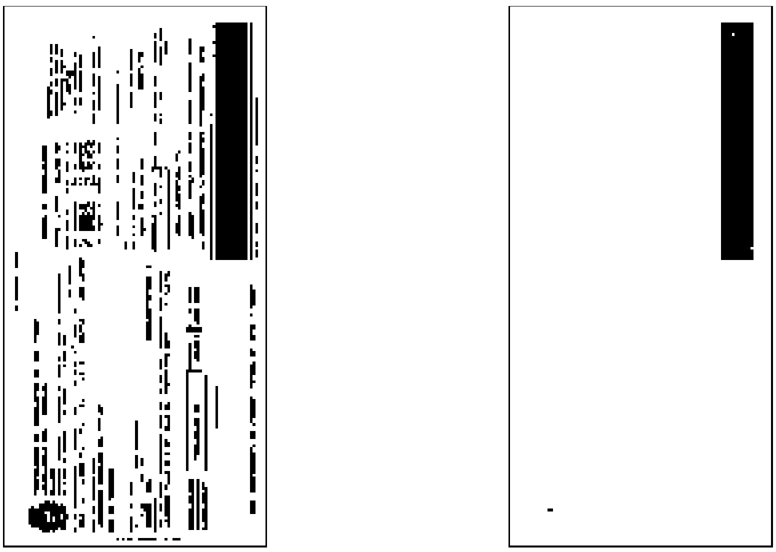


Figura 9.17: Imagens de treinamento relativos à localização de código de barras - dimensão da imagem 202 × 98 pixels.



a b

Figura 9.18: Resultado da localização de código de barras.

9.7 Identificação de Fissuras em Metais

O problema de identificação de fissuras em imagens microscópicas de metais é um problema clássico ([Sch89]). As fissuras são caracterizadas por regiões desconexas entre os filamentos observáveis nas imagens, como a ilustrada na figura 9.19. Tal estudo permite, por exemplo, avaliar a resistência de um metal.

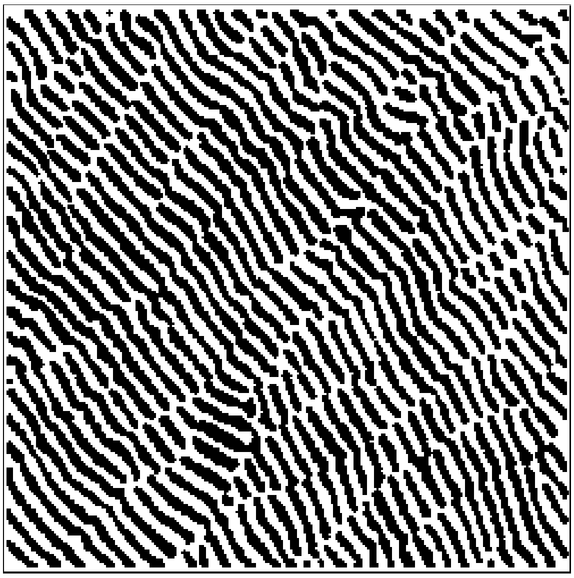


Figura 9.19: Imagem de metal - dimensão da imagem 250×250 pixels.

Se considerarmos este problema no contexto de aprendizado proposto, realizar o treinamento na escala original da imagem requer uma janela relativamente grande, dadas as características dos objetos na imagem. Realizamos alguns testes, porém nenhum resultado satisfatório foi obtido, uma vez que a única imagem disponível era esta, e o número de exemplos é muito pequeno relativamente ao tamanho da janela utilizada.

Uma abordagem encontrada foi a realização de um pré-processamento na escala original da imagem, seguida de redução de escala, sobre o qual foi realizado o treinamento. O pré-processamento consistiu de algumas transformações na imagem original, para alargar os espaços existentes entre as pontas dos filamentos, de forma a evitar que dois filamentos separados se unissem ao se reduzir a escala da imagem. A figura 9.20 mostra as imagens de treinamento, com escala reduzida.

A tabela 9.10 descreve o experimento realizado com as imagens de treinamento ilustrados na figura 9.20.

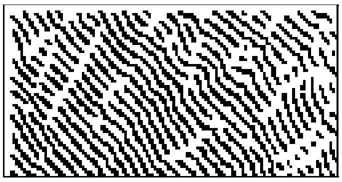
 

Figura 9.20: Imagens de treinamento relativos à identificação de fissuras em metais.

Parâmetro	valor ou medida
janela	5×5
m	$7,260$
exemplos distintos	$3,812$
exemplos contraditórios	65
exemplos negativos	$3,519$
exemplos positivos	293
algoritmo	ISI-3
tempo	$30s$
base	127

Tabela 9.10: Experimento relativo à identificação de fissuras em metais.

Na figura 9.21 mostramos a imagem no qual foi aplicado a base obtida, o resultado obtido, e a sobreposição das duas.

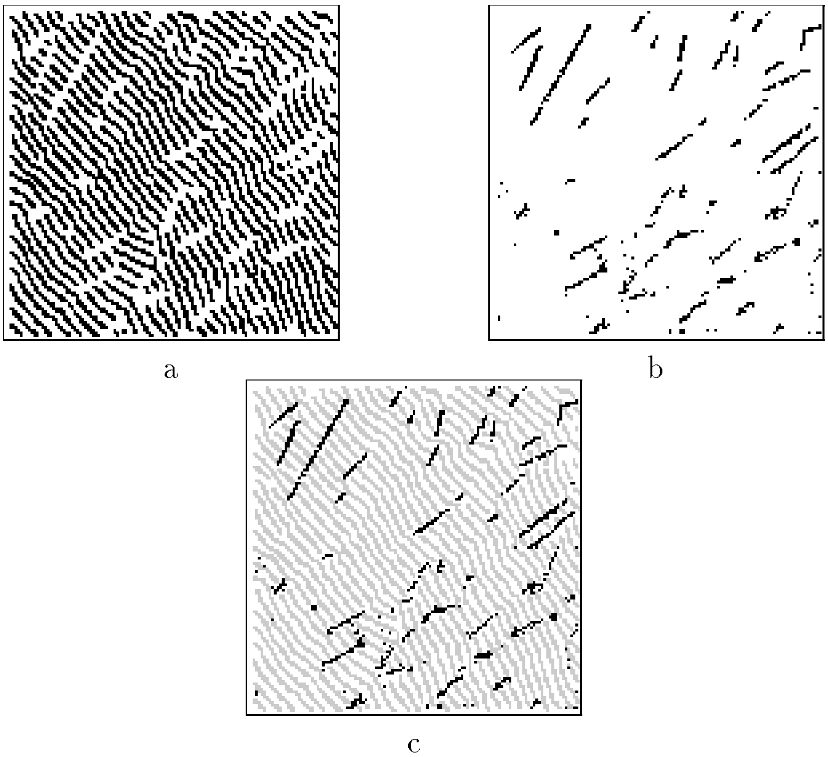


Figura 9.21: Resultado relativo à identificação de fissuras em metais.

9.8 Reconhecimento de Padrões

Os problemas de reconhecimento de padrões são problemas clássicos. Existem várias técnicas utilizadas para resolver estes problemas ([GW92, TG74]). Encontramos alguns trabalhos como [DH73] e [TG74] que apresentam alguns métodos de reconhecimento automático de padrões baseado, por exemplo, em funções de decisão.

9.8.1 Introdução

Um padrão pode ser uma estrutura, uma propriedade, um comportamento, uma forma ou aspecto que caracterizam elementos dentro de um determinado contexto. Uma vez que existam padrões suficientes para caracterizar os diversos elementos do contexto, o desafio que se segue é o reconhecimento de elementos de interesse pela observação de padrões que os caracterizam. Em análise de imagens, os padrões mais comuns são formas ou texturas. Existem várias aplicações, tais como, o reconhecimento de células doentes em função de seu tamanho anormal, ou o reconhecimento de áreas que representam a plantação de determinado produto agrícola baseado na textura observada em imagens de satélites, o reconhecimento de letras 'a' em um texto baseado estritamente no seu formato, o reconhecimento de defeitos em produtos numa linha de produção, etc.

Nesta seção, formalizaremos os problemas de reconhecimento de padrões que são baseados em análise de formas dentro do contexto da *MM*.

Um **padrão** $\mathcal{T}$ em E é uma coleção de subconjuntos de E. Um elemento $X \in \mathcal{T}$ é denominado um **elemento de padrão** $\mathcal{T}$.

O problema de reconhecimento de padrões pode ser formulado da seguinte forma.

Seja I um conjunto de índices e seja $\{\mathcal{T}_i : i \in I\}$ uma coleção de padrões tal que $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ para $i \neq j$, $i, j \in I$. O problema de reconhecimento de padrões pode ser formulado através da seguinte pergunta : dado um elemento $X \in \bigcup\{\mathcal{T}_i : i \in I\}$, a qual dos padrões pertence X ?

Uma coleção de operadores i.t. localmente definidos $\{\psi_i \in \Psi_{W_i} : i \in I, W_i \subseteq E\}$ tal que para qualquer i, $i \in I$, ψ_i satisfaz

$$\psi_i(X) \neq \emptyset,\ \forall\ X \in \mathcal{T}_i$$

e

$$\psi_i(X) = \emptyset,\ \forall\ X \in \{\mathcal{T}_j : j \in I, j \neq i\}$$

pode ser utilizada para resolver este problema.

O operador ψ_i é denominado **marcador** do padrão $\mathcal{T}_i$ e é localmente definido pela janela W_i. A menor janela W_i que satisfaz as condições acima determina a dimensão do problema de reconhecimento do padrão $\mathcal{T}_i$.

Observe que todos os marcadores são operadores anti-extensivos. Portanto, quando consideramos o problema no contexto de aprendizado, pode-se utilizar o intervalo $[\{o\}, W]$ para inicializar o algoritmo de aprendizado ISI. A função de perda para os problemas de reconhecimento de padrões é dado por

$$l_h(x,b) = \begin{cases} 0 & \text{se } h(x) = b = 0 \ , \\ 0 & \text{se } h(x) = b = 1 \text{ e } p(b = 1/x) = 1 \\ 1 & \text{caso contrário} \end{cases}$$

O critério de decisão que minimiza a perda de (x, b) é dado por

$$h(x) = \begin{cases} 1 & \text{se } b = 1 \text{ e } p(b = 1/x) = 1 \\ 0 & \text{caso contrário} \end{cases}$$

9.8.2 Reconhecimento de Dígitos

Neste experimento, geramos dígitos de 0 a 9, em 10 fontes distintas, conforme ilustra a figura 9.22. Selecionamos manualmente todos os dígitos 8 desta imagem (figura 9.23) e efetuamos o treinamento sobre este par de imagens. O objetivo é obter um marcador para os dígitos 8, ou seja, reconhecer os padrões que caracterizam os dígitos 8.

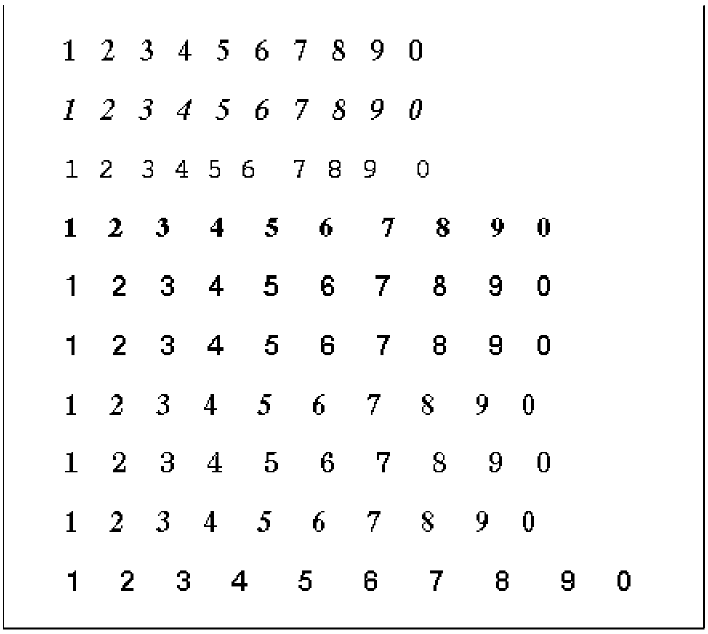


Figura 9.22: Imagem com dígitos de “0” a “9”.

A tabela 9.11 descreve o experimento realizado utilizando-se uma janela 3×3.

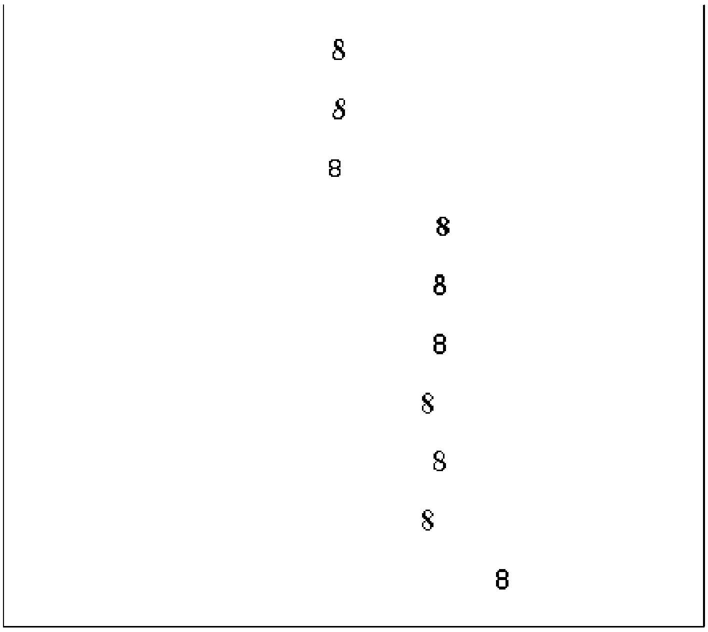


Figura 9.23: Imagem apenas com os dígitos "8".

A base obtida, com 7 elementos, foi aplicada sobre a mesma imagem de treinamento e resultou nos marcadores (pontos pretos, sobrepostos sobre a imagem na qual foi aplicada) expressos na imagem da figura 9.24.

Parâmetro	valor ou medida
janela	3×3
m	$168, 345$
exemplos distintos	268
exemplos contraditórios	80
exemplos negativos	257
exemplos positivos	11
algoritmo	ISI-3
tempo	0.8s
memória	16
base	7

Tabela 9.11: Reconhecimento de dígitos - janela 3×3.

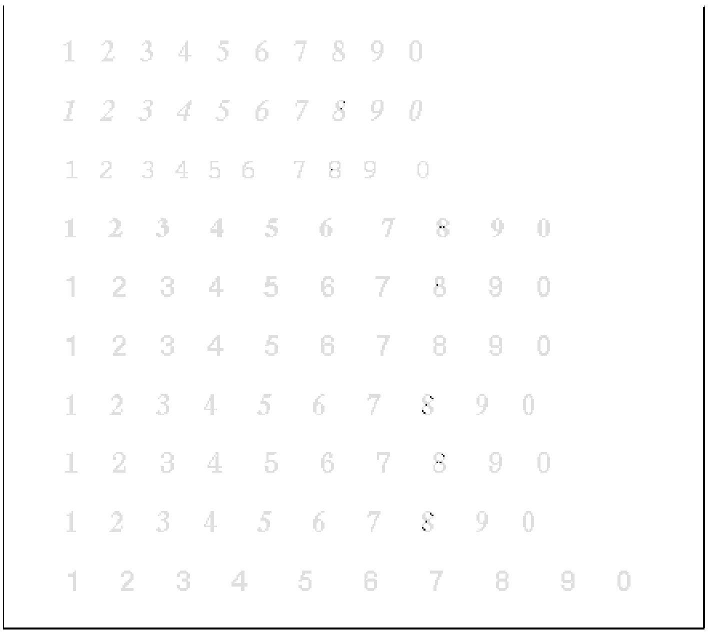


Figura 9.24: Marcadores obtidos com uma janela 3×3.

Observe que este operador não foi capaz de reconhecer todos os dígitos 8 (Deixou de marcar os dígitos "8" das linhas 1, 6 e 9). Isto indica-nos que o tamanho da janela considerada não foi suficiente para o problema considerado. Portanto, realizamos um outro experimento com uma janela maior, conforme descrevemos na tabela 9.12.

Parâmetro	valor ou medida
janela	4×3
m	$167,910$
exemplos distintos	756
exemplos contraditórios	124
exemplos negativos	719
exemplos positivos	37
algoritmo	ISI-3
tempo	0.81s
memória	35
base	24

Tabela 9.12: Reconhecimento de dígitos - janela 4×3.

Para o experimento com a janela 4×3 foi obtida uma base maior, com 24 elementos e, conforme ilustra a figura 9.25, a janela foi suficiente para gerar marcadores para todos os dígitos 8.

A *reconstrução* (veja, por exemplo, [BB94], página 139) da imagem através dos marcadores obtidos extrai todos os dígitos marcados, desde que os dígitos não estejam conexos uns aos outros. A figura 9.26 mostra a sobreposição da imagem reconstruída sobre a imagem original.

Obviamente, ao se aplicar o operador aprendido sobre a própria imagem de treinamento o resultado obtido deve ser perfeito no caso em que não há exemplos contraditórios na amostra. Entretanto, quando a janela não é suficientemente grande para caracterizar determinados padrões de interesse, existem exemplos contraditórios, e portanto o resultado não é perfeito. A medida que se aumenta o tamanho da janela, o número de exemplos contraditórios tende a diminuir e, conseqüentemente, o resultado aproxima-se do ideal.

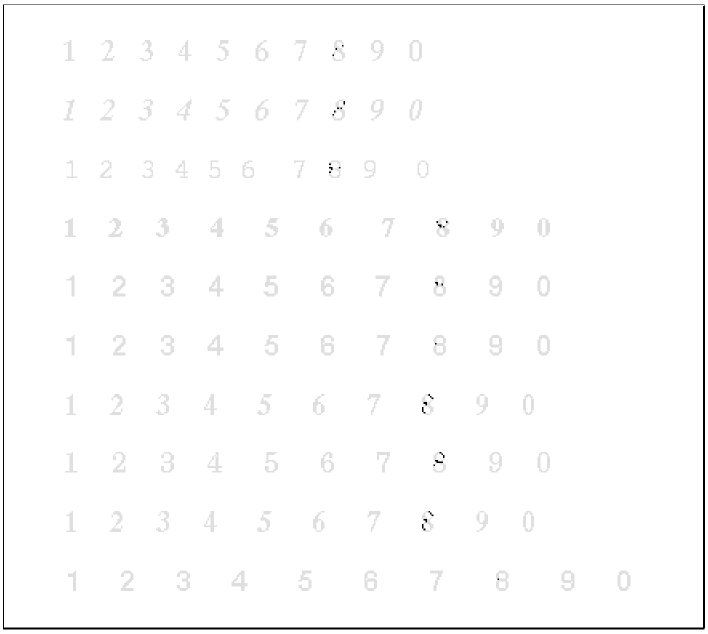


Figura 9.25: Marcadores obtidos com uma janela 4×3.

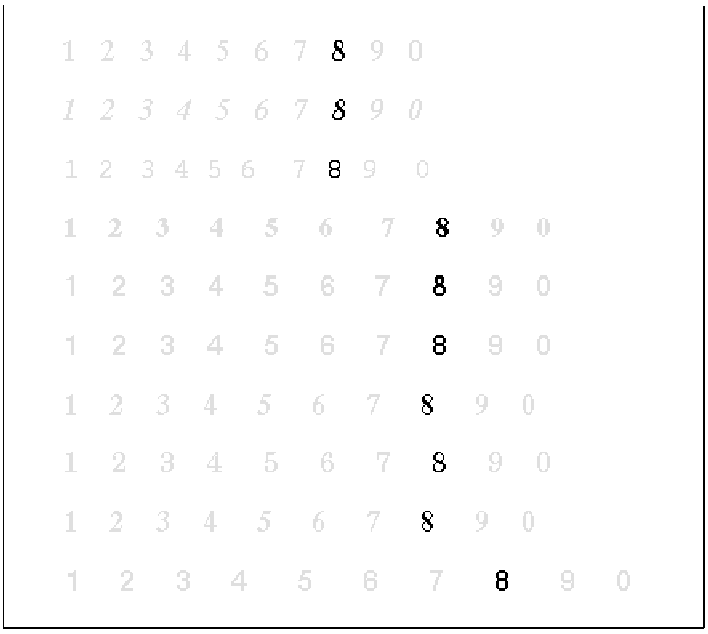


Figura 9.26: Reconstrução a partir dos marcadores obtidos com uma janela 4×3.

9.8.3 Reconhecimento de Letras

Neste exemplo consideramos trechos de texto capturados do monitor de vídeo. O treinamento foi realizado sobre as imagens da figura 9.27. Este exemplo ilustra uma aplicação do sistema para o reconhecimento de letras "a" minúsculas.

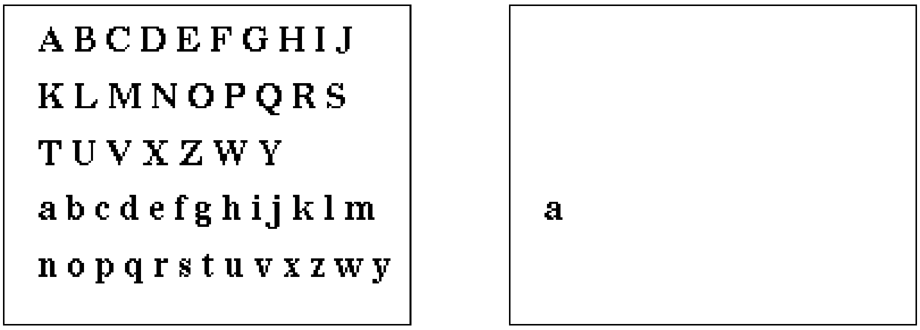


Figura 9.27: Imagens de treinamento para o reconhecimento de letra "a".

A tabela 9.13 descreve o treinamento realizado.

Parâmetro	valor ou medida
janela	3×3
m	$24,426$
exemplos distintos	273
exemplos contraditórios	25
exemplos negativos	271
exemplos positivos	2
algoritmo	ISI-3
tempo	0.8s
memória	14
base	2

Tabela 9.13: Reconhecimento de letras "a".

A base obtida, com 2 elementos, conforme relação a seguir, foi aplicada sobre a imagem da figura 9.28a e os marcadores obtidos foram sobrepostos à imagem original (figura 9.28b). A reconstrução a partir dos marcadores foi sobreposta sobre a imagem original para melhor visualizarmos o resultado final, na figura 9.28c.

$$
1)\quad \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{array} \qquad \begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array} \qquad\qquad 2)\quad \begin{array}{ccc} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{array} \qquad \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}
$$

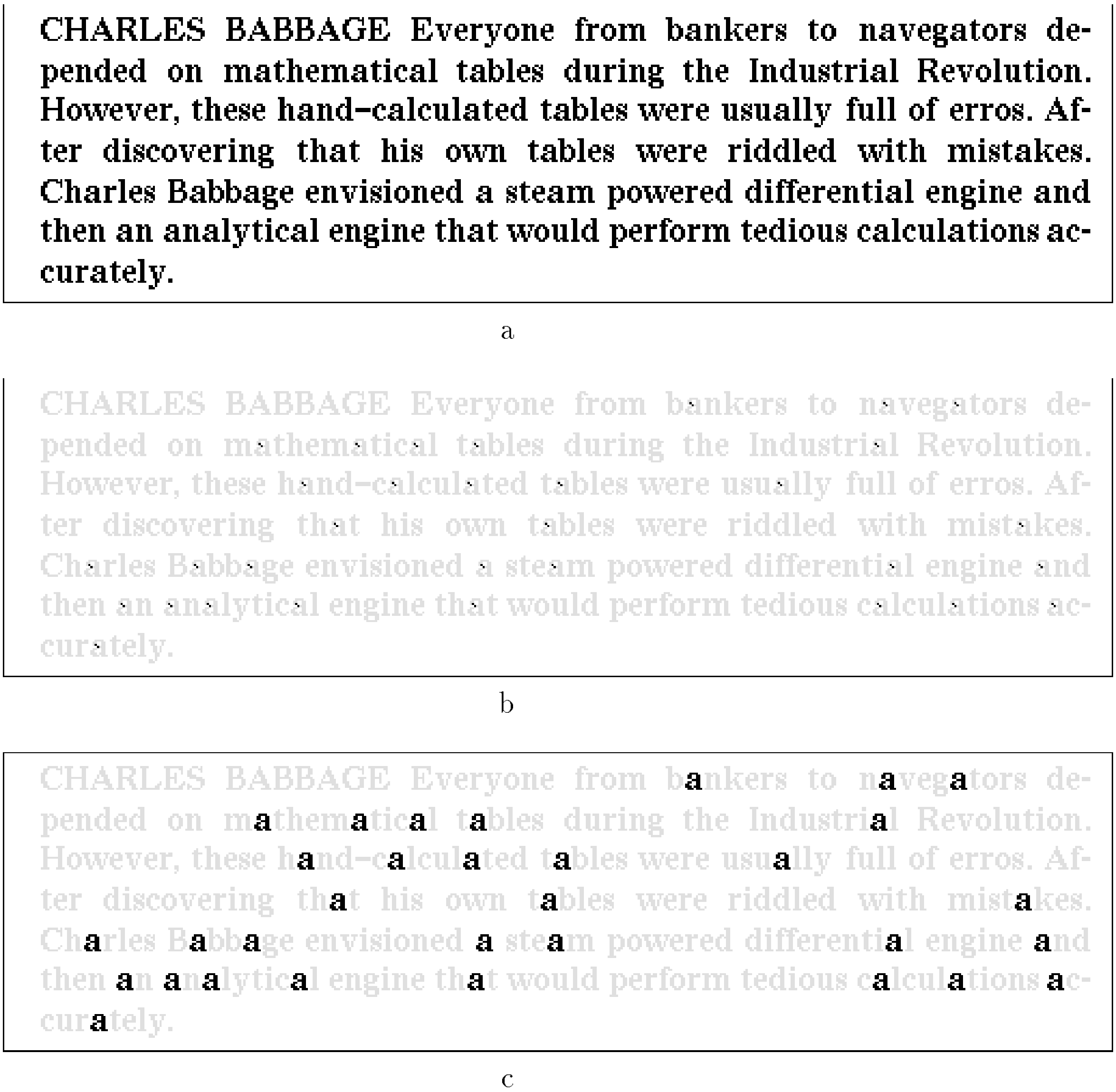


Figura 9.28: Resultado relativo ao reconhecimento de letras “a”.

Capítulo 10

Reconhecimento de Caracteres Impressos

O reconhecimento de caracteres impressos é uma atividade importante pois permite que textos impressos (livros, documentos, enciclopédias) possam ser codificados de modo a serem armazenados em um meio mais compacto (tais como fitas e discos magnéticos ou discos óticos) e possam ser manipulados eficientemente através de vários recursos de computação disponíveis.

O problema de reconhecimento de caracteres impressos é mais complexo do que muitos exemplos tratados no capítulo anterior, pois o processo de aquisição de imagens está sujeito a alguns tipos de ruído.

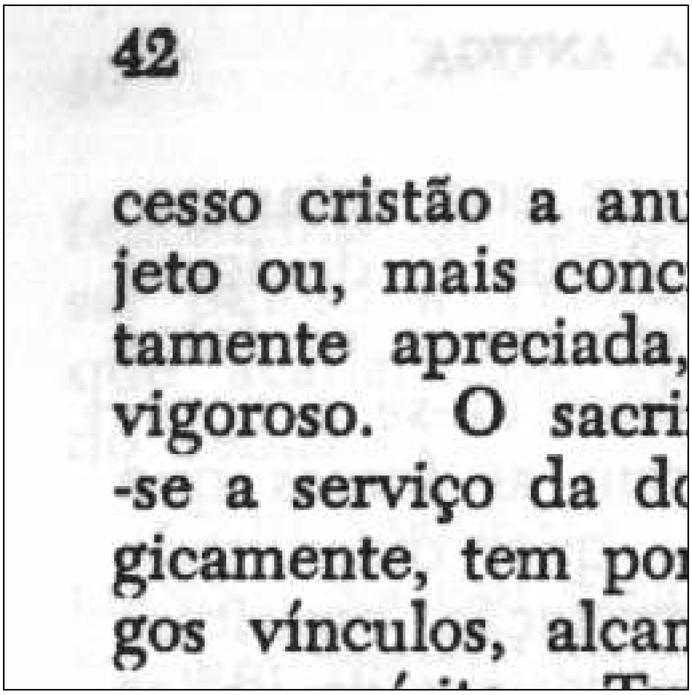
10.1 Reconhecimento de Caracteres Impressos

Realizamos vários experimentos de reconhecimento de caracteres impressos de dois livros velhos, escritos em português (veja também [BTdST96]). Para entendermos os tipos de ruídos envolvidos no problema, descrevemos inicialmente o processo de aquisição de imagens.

10.1.1 Coleta de imagens

Foram geradas as imagens correspondentes a 15 páginas escolhidas aleatoriamente de cada um dos livros, a partir de um digitalizador óptico ("scanner"). Utilizamos a resolução de 200 dpi (pontos por polegada). As imagens obtidas em níveis de cinza foram segmentadas por operadores morfológicos para transformá-las em imagens binárias (na qual os pixels de valor 1 correspondem aos caracteres e os de valor 0 correspondem ao fundo da imagem). Estas imagens foram ainda reduzidas em sua escala para 50% da sua dimensão original. Portanto estão envolvidos três tipos de ruído neste processo : 1) durante a digitalização, 2) na segmentação e 3) na redução de escala.

As figuras 10.1, 10.2 e 10.3 ilustram as imagens envolvidas neste processo. A figura 10.1 corresponde à imagem obtida pelo digitalizador óptico, a figura 10.2 corresponde à segmentação da primeira e a figura 10.3 corresponde à imagem segmentada reduzida.

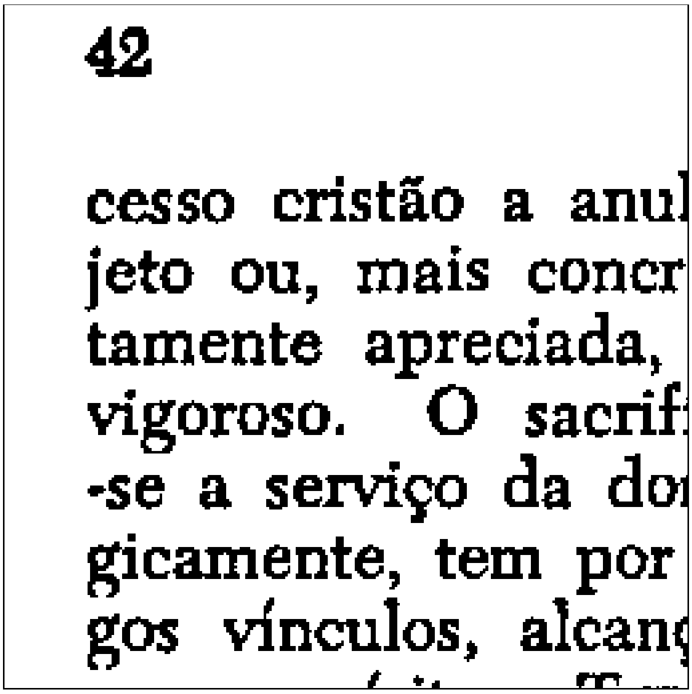


42

cesso cristão a anu
jeto ou, mais conc
tamente apreciada,
vigoroso. O sacri
-se a serviço da d
gicamente, tem po
gos vínculos, alcan

Figura 10.1: Imagem capturado por um digitalizador óptico.

De cada imagem segmentada foram extraídas manualmente (através de um editor de imagens) todas as ocorrências de letras "a" e letras "s" minúsculas. As 15 páginas foram separadas em dois grupos : imagens de treinamento e imagens de validação. As imagens de treinamento foram utilizadas para realizar o treinamento e as imagens de validação foram utilizadas para avaliar a precisão do operador aprendido. Todos os treinamentos foram realizados sobre as imagens reduzidas.



42

cesso cristão a anu
jeto ou, mais concr
tamente apreciada,
vigoroso. O sacrif
-se a serviço da do
gicamente, tem por
gos vínculos, alcan

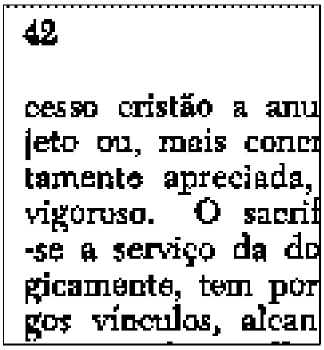
Figura 10.2: Imagem segmentada.

10.1.2 Descrição dos Resultados Obtidos

A seguir apresentamos duas tabelas, tabelas 10.1 e 10.2, que descrevem, respectivamente, vários experimentos realizados para o reconhecimento das letras "s" e "a" do livro 1.

Cada uma das linhas das tabelas 10.1 e 10.2 correspondem a um experimento. Em cada uma destas tabelas *janela* indica as dimensões da janela utilizada; *Número de exemplos* indica o tamanho da amostra de treinamento (independente de exemplos repetidos); *tipo do algoritmo* indica a variante do algoritmo ISI utilizado; *Tamanho da base* indica o tamanho da base do operador aprendido; *tempo de treinamento* indica o tempo de processamento do algoritmo ISI, dado em termos de horas (h), minutos (m) e segundos (s); *erro relativo* indica a porcentagem de erros cometido pelo operador aprendido.

Os exemplos contraditórios foram todos considerados negativos. O erro relativo foi calculado sobre as imagens de validação, levando-se em conta dois tipos de erro : erro por falta e erro por excesso. Os erros por falta são caracterizados quando o operador aprendido não marca uma letra que deveria ser reconhecida e os erros por excesso são caracterizados quando o operador marca uma letra que não corresponde à letra que deveria ser reconhecida. O erro relativo é a proporção da soma destes dois tipos de erro em relação


42

cesso cristão a anu
jeto ou, mais conc
tamento apreciada,
vigoroso. O sacrif
-se a serviço da do
gicamente, tem por
gos vínculos, alcan

Figura 10.3: Imagem reduzida em escala.

ao total de caracteres sobre as quais foram aplicadas a validação.

janela	número de exemplos	tipo do algoritmo	tamanho da base	tempo de treinamento	erro relativo (%)
5×5	270,267	ISI-2	1,560	5h20m	6.2
5×5	79,049	ISI-3	416	20m53s	7.66
7×7	79,040	ISI-2	1,348	1h46m	14.85
7×7	79,040	ISI-3	284	25m	5.12

Tabela 10.1: Experimentos para o reconhecimento da letra “s” no livro 1.

Em relação a estes experimentos, observamos claramente que o número de exemplos considerado no treinamento afeta diretamente a qualidade do resultado final (veja linhas 1, 2 e 4 na tabela 10.2). Entretanto, quanto maior o número de exemplos de treinamento, maior é o tempo gasto pelo algoritmo de aprendizado (linhas 1, 2 e 4 da tabela 10.2). Portanto, aumentar indiscriminadamente o número de exemplos não parece ser uma boa solução para melhorar o resultado. Além disso, pode-se observar nas linhas 1 e 2 da tabela 10.1 que o ganho relativo em se aumentar o número de exemplos não é muito significativo.

Podemos observar também, nas linhas 2 e 4 da tabela 10.1, ou linhas 2 e 6 ou 3 e 5 da tabela 10.2, que não há uma relação evidente entre o tamanho da janela e o erro relativo.

Além disso, conforme vimos no capítulo 6, as variantes do algoritmo ISI produzem bases de tamanho diferente para uma mesma amostra de treinamento (basicamente diferem quanto à atribuição de valor aos don't cares). Conseqüentemente, o resultado final relativo aos operadores gerados por diferentes algoritmos podem diferir, como podemos verificar

janela	número de exemplos	tipo de algoritmo	tamanho da base	tempo de treinamento	erro relativo (%)
7×7	270,192	ISI-2	5,059	267h12m	15.5
7×7	79,040	ISI-2	2,311	19h20m	25.56
7×7	79,040	ISI-3	644	2h47m28s	10.4
7×7	37,201	ISI-2	1,447	5h18m	35.31
9×9	79,019	ISI-3	551	3h45m	12.75
9×9	79,019	ISI-2	2,798	14h42m	31.45

Tabela 10.2: Experimentos para o reconhecimento da letra “a” no livro 1.

nas linhas 3 e 4 da tabela 10.1 ou nas linhas 2 e 3 ou linhas 5 e 6 da tabela 10.2.

10.1.3 Experimentos com Múltiplos Estágios de Treinamento

Nos experimentos da seção anterior, notamos que quase todos os erros verificados são erros por excesso. Ou seja, muitos caracteres que não deveriam ser reconhecidos foram marcados. Esta observação, sugeriu-nos a aplicação da estrutura seqüencial do sistema, descrita no capítulo 8. Neste caso, um segundo operador seria aprendido a partir das imagens transformadas pelo primeiro operador. Este segundo operador age como uma espécie de filtro, cuja finalidade é refinar o resultado anterior. Esta mesma idéia pode ser aplicada sucessivamente até que alguma estabilidade seja atingida. O número de treinamentos realizados determina o número de estágios do experimento.

As tabelas 10.3 e 10.4 descrevem resultados referentes a vários experimentos realizados com treinamento de múltiplos estágios. A coluna *janela no estágio 1* indica o tamanho da janela utilizada no primeiro estágio; *número total de exemplos* indica a soma do número total de exemplos utilizados (sem levar as repetições em consideração); *número de estágios* indica o número de estágios do treinamento; *tamanho total da base* indica a soma do tamanho das bases; *tempo total de treinamento* indica a soma do tempo total de treinamento; *erro relativo* indica o erro relativo do resultado final.

O conjunto de imagens de treinamento foi subdividido em grupos, de modo que as imagens utilizadas em um estágio de treinamento não fossem mais utilizadas nos demais estágios. O tamanho da janela foi reduzido por 2 pontos, tanto na largura como na altura, de um estágio para o estágio subsequente. Em todos os treinamentos dos experimentos com múltiplos estágios foi utilizado o algoritmo ISI-3.

Os experimentos com treinamentos de múltiplo estágio permitiram-nos observar alguns resultados interessantes.

janela no estágio 1	número total de exemplos	número de estágios	tamanho total da base	tempo total de treinamento	erro relativo (%)
5×5	79,049	1	416	20m53s	7.66
5×5	86,111	2	429	20m54s	1.31
7×7	79,040	1	284	24m31s	5.12
7×7	83,237	2	354	24m49s	0.49
7×7	87,288	3	388	24m50s	0.35

Tabela 10.3: Reconhecimento da letra "s" no livro 1 - treinamento de múltiplos estágios.

janela no estágio 1	número total de exemplos	número de estágios	tamanho total da base	tempo total treinamento	erro relativo (%)
7×7	79,040	1	644	2h47m28s	10.4
7×7	88,333	2	726	2h47m55s	1.38
7×7	96,532	3	762	2h47m56s	0.50
9×9	79,019	1	551	3h45m	12.75
9×9	88,275	2	700	3h47m52s	0.80
9×9	96,121	3	760	3h48m13s	0.47
9×9	103,530	4	781	3h48m14s	0.38

Tabela 10.4: Reconhecimento da letra "a" no livro 1 - treinamento com múltiplos estágios.

Observamos inicialmente que há uma considerável redução do erro relativo do treinamento de 1 estágio para o de 2 estágios (compare figuras 10.4 e 10.5[1]), enquanto o acréscimo no tempo de treinamento e no tamanho da base não é significativo relativamente a esta redução. Uma explicação informal para este fato pode ser expressa através do próximo parágrafo.

Após a aplicação do primeiro operador (aprendido no primeiro estágio de treinamento), observa-se que a imagem resultante possui uma concentração de marcas sobre a letra-alvo (aqueles que devem ser reconhecidos) e alguns marcadores esparsos (erros por excesso) sobre as demais letras (veja figura 10.4). Nota-se que, embora a forma da letra-alvo nem sempre se preserve após a aplicação do operador, esta concentração de pontos é suficiente para distinguir as letras-alvo das demais. O segundo operador (obtido do treinamento realizado no segundo estágio) age como um filtro sobre esta imagem, eliminando a maior

[1]Nestas figuras, a parte preta representa os marcadores (isto é, o resultado do operador aprendido. Os marcadores foram sobrepostos sobre a imagem anterior à transformação.

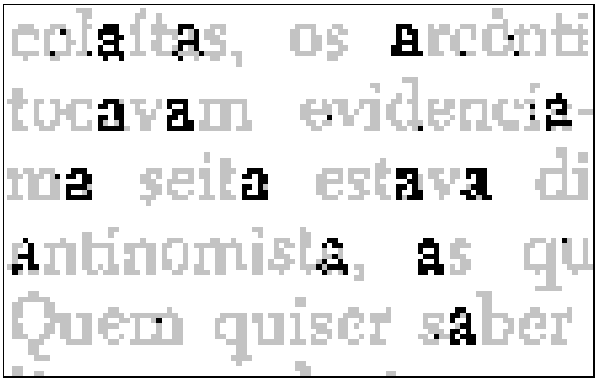


Figura 10.4: Resultado do primeiro estágio.

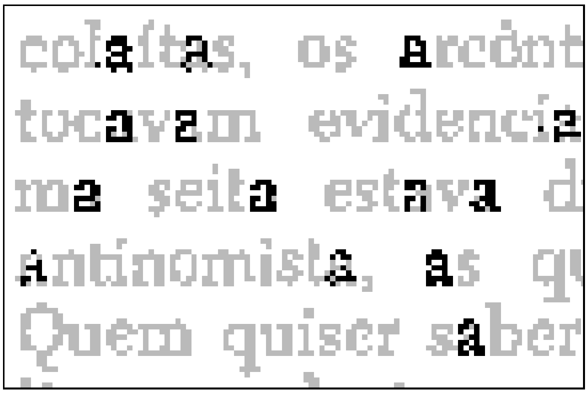


Figura 10.5: Resultado do segundo estágio.

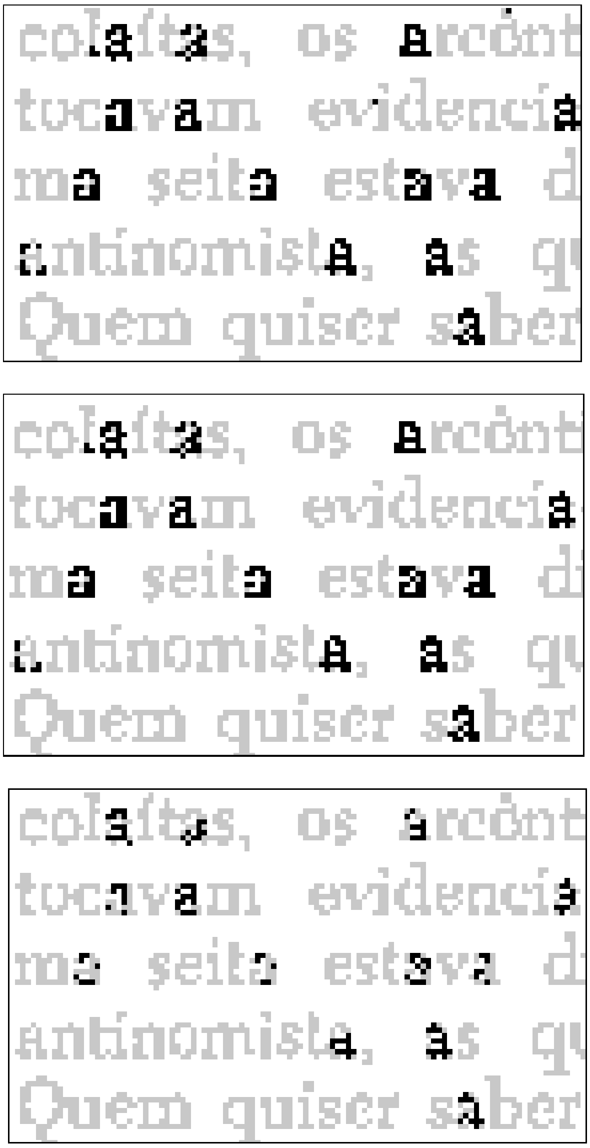
parte dos marcadores esparsos. A partir do terceiro estágio, esta característica já não é mais tão acentuada, provavelmente porque o número de pontos esparsos foi reduzido significativamente pelo segundo operador.

Experimentos com o segundo livro mostraram que a partir de um determinado estágio, os erros por falta começam a ser superiores aos erros por excesso. Este fato impõe um limite na aplicação sequencial deste método.

Estas observações sugerem que as aplicações sucessivas de estágios de treinamento fazem o erro por excesso decrescer até um ponto de equilíbrio no qual o erro relativo atinge o mínimo, e a partir deste ponto os estágios subsequentes começam a adicionar erros por falta, acarretando o crecimento do erro relativo.

Uma outra observação, diz respeito ao tamanho da janela utilizada no primeiro estágio de treinamento. Pelas linhas 2 e 4 da tabela 10.3 ou linhas 2 e 5 da tabela 10.4 verifica-se que o tamanho da janela utilizada no primeiro estágio está diretamente associado à qualidade do resultado final.

As figura 10.6 mostra o resultado dos três estágios de treinamento, referentes ao livro 1, iniciados com uma janela 7×7 para o reconhecimento da letra "a" . Observe que o número de marcadores diminui de um estágio para outro.



1

2

3

Figura 10.6: Os três estágios de reconhecimento de letras "a" do livro 1.

A figura 10.7 mostra o resultado dos quatro estágios (iniciados com uma janela 9×9), referente ao livro 1, para reconhecimento da letra "a".

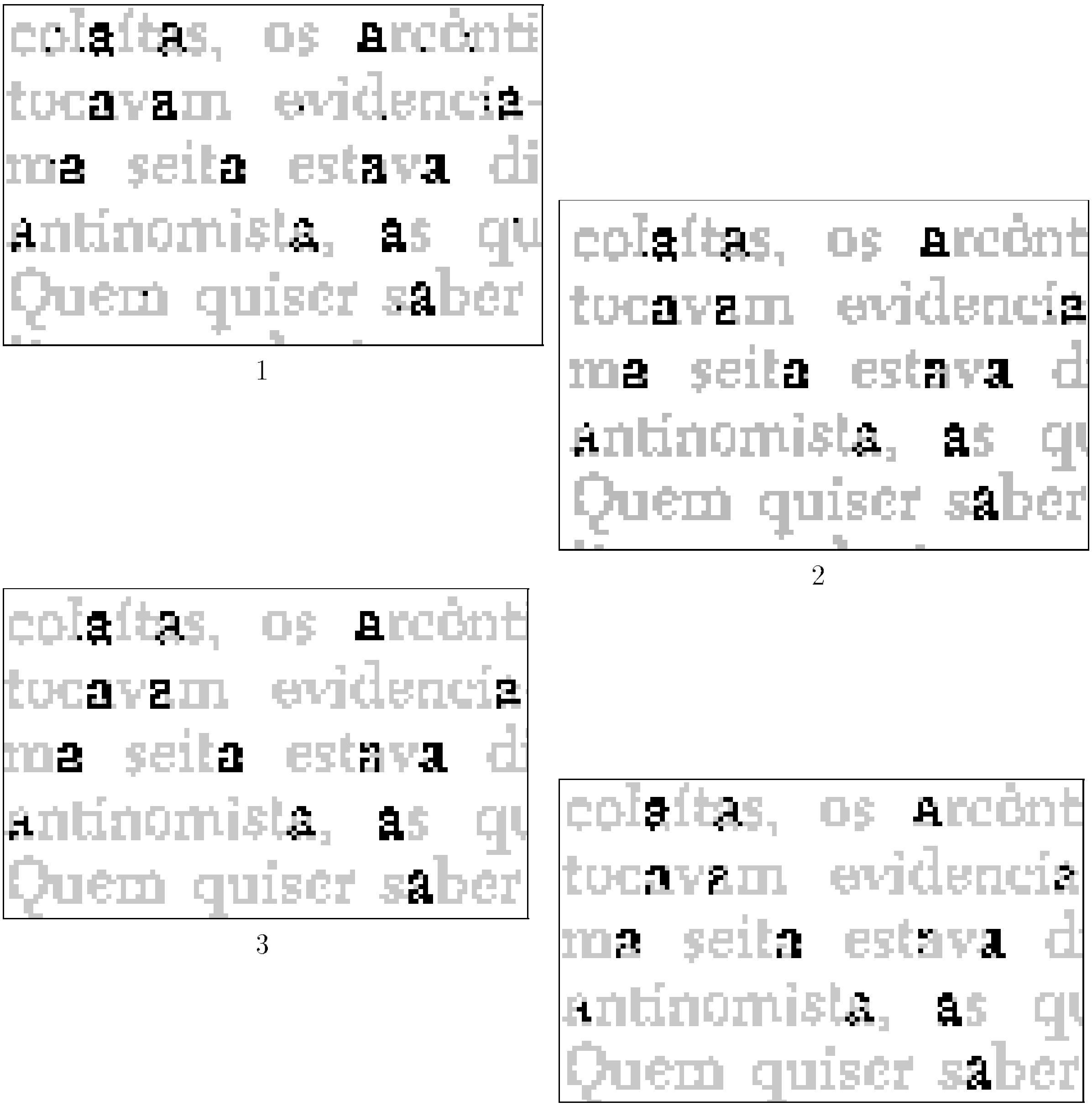
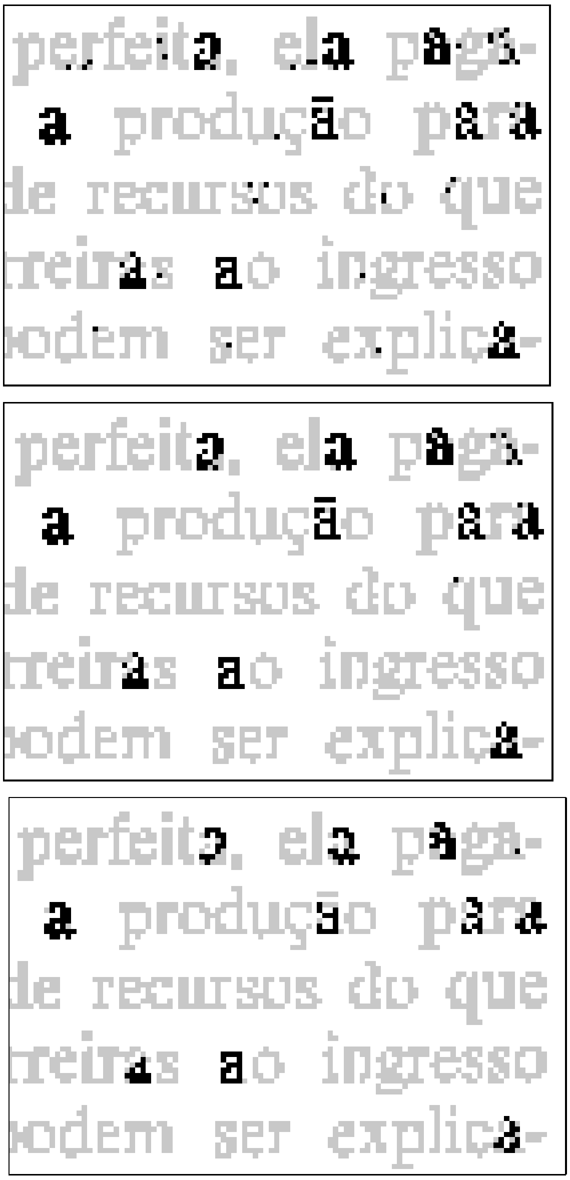


Figura 10.7: Os quatro estágios de reconhecimento de letras "a" do livro 1.

A figura 10.8 mostra o resultado dos três estágios, iniciados com uma janela 7×7, referente ao livro 2, para reconhecimento da letra "a".



1

2

3

Figura 10.8: Os três estágios de reconhecimento de letras "a" do livro 2.

A figura 10.9 mostra o resultado dos quatro estágios 9iniciados com uma janela 9×9), referente ao livro 2, para reconhecimento da letra "a".

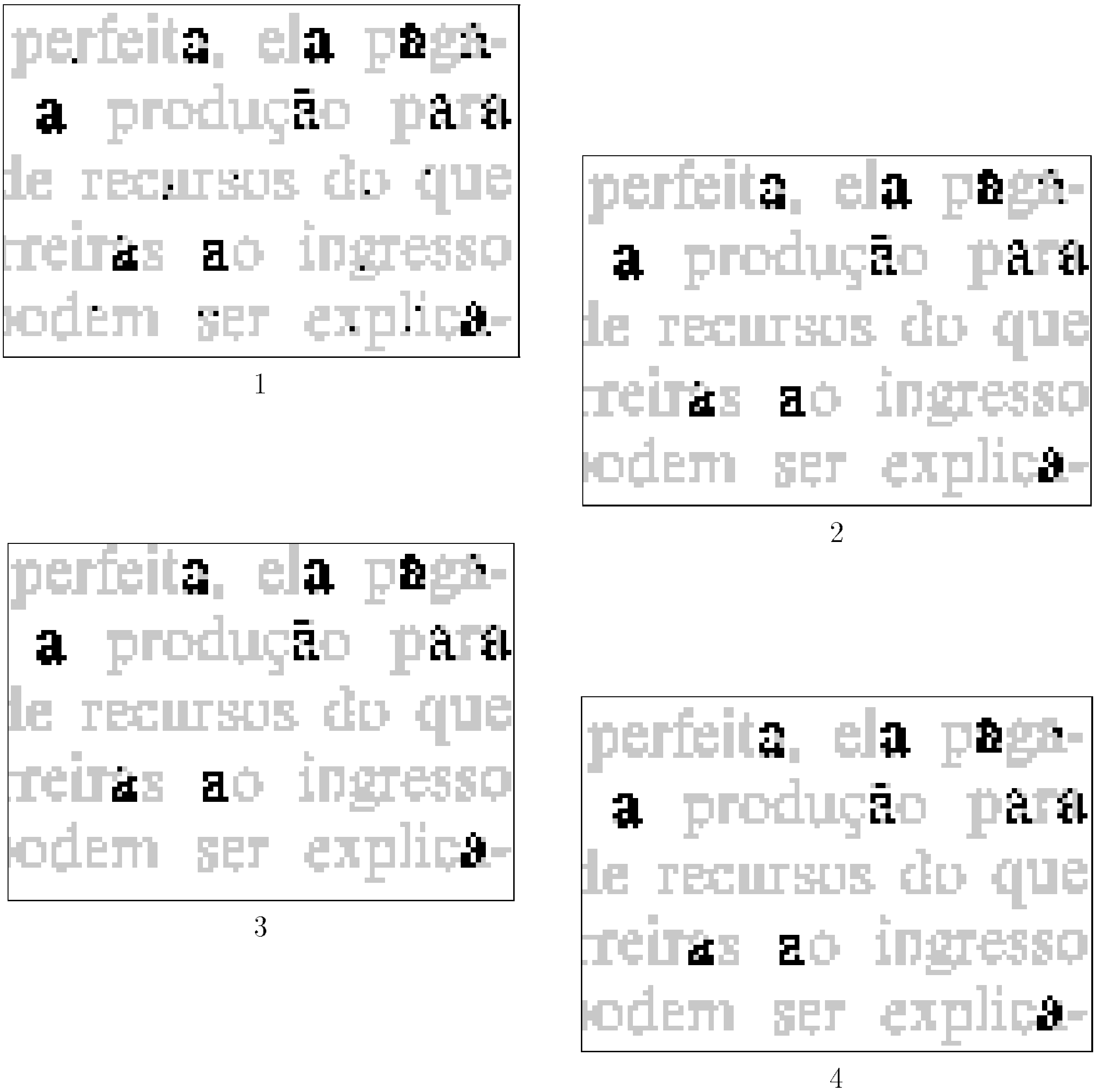
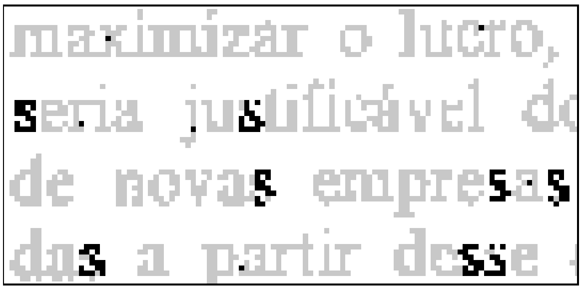
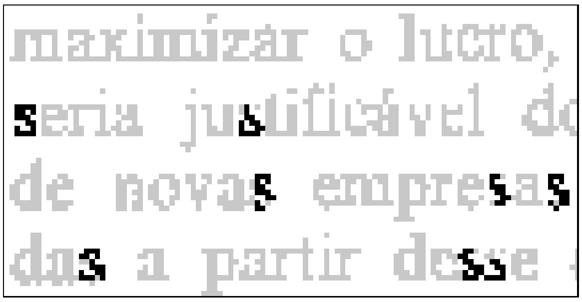


Figura 10.9: Os quatro estágios de reconhecimento de letras "a" do livro 2.

A figura 10.10 mostra o resultado do primeiro e terceiro estágios de treinamento, iniciados com uma janela 7×7, referentes ao livro 2, para reconhecimento da letra “s”.



1



3

Figura 10.10: Resultado dos estágios 1 e 3 para reconhecimento de letras “s” do livro 2.

10.2 Análise dos Resultados

Apesar da existência de certas limitações (como o tempo de treinamento necessário), os resultados obtidos são bastante encorajadores. Além disso, a aplicação da estrutura seqüencial mostrou-se um alternativa viável para contornar esta limitação, oferecendo um resultado melhor, sem no entanto aumentar demasiadamente o número de exemplos de treinamento. Sem o uso da estrutura seqüencial, o mesmo resultado só poderia ser atingido com um conjunto de exemplos de treinamento muito maior. Verificamos, no entanto que, para se obter algum ganho significativo no resultado, a quantidade de exemplos adicionais necessários pode ser extremamente alta.

Outra vantagem da aplicação sequencial é que ela gera os operadores numa forma sequencial, ou seja, uma frase mais curta do que a gerada pelo treinamento de um único operador (com maior número de exemplos).

Uma outra questão que merece ser analisada está relacionada ao número de exemplos utilizados nos treinamentos. Podemos dizer que os resultados obtidos (mais de 99.5% de acerto) são excepcionais. No entanto, observamos uma distância significativa entre o limite teórico (m_0) e o número real utilizado. Para termos uma idéia desta distância, analisemos o caso de uma janela 7×7, para $\epsilon = \delta = 0.25$, utilizando o limite teórico para o modelo PAC básico. O limite teórico neste caso é dado por

$$m(\delta, \epsilon) = \frac{1}{0.25} ln(\frac{2^{2^{49}}}{0.25}) \sim 10^{15}$$

enquanto que para um experimento utilizando $m = 270,000$ obtivemos a mesma precisão.

Acreditamos que esta distância pode estar relacionada com o fato de estarmos considerando, para o cálculo do limite teórico, todas as $2^{2^{|W|}}$ possíveis configurações, quando na prática as configurações observadas por uma janela estão restritos a um suconjunto $\mathcal{A} \subseteq \mathcal{P}(W)$. Neste caso, o tamanho do espaço de hipóteses seria dado por $2^{|\mathcal{A}|}$.

No entanto, modelar o contexto no qual se encontram as configurações não é simples.Neste ponto, parece-nos que os "random sets" são recursos importantes.

A conclusão que podemos extrair dos experimentos realizados é que a abordagem considerada representa uma forma sólida e abrangente para a resolução de vários problemas de processamento de imagens binárias.

Paralelamente, verificamos que existem muitas questões que ainda não podem ser respondidas pois necessitam de investigações mais profundas. Abordaremos estas questões na conclusão deste texto.

Capítulo 11

Conclusão

11.1 Um Apanhado Geral

Com base no fato de que, quando restritos ao domínio das imagens binárias, vários problemas de análise de imagens podem ser resolvidos usando a *MM* para subconjuntos e, em particular, por uma classe formada por aqueles que são invariantes por translação e localmente definidos por uma janela W, estudamos a sua decomposição canônica e investigamos sua relação com o conjunto das funções Booleanas de n variáveis onde n é a cardinalidade da janela W (capítulo 3).

Face à dificuldade de geração ou criação de um operador morfológico adequado para resolver um determinado problema, sugerimos um modelo para programação automática de MMach's, isto é, um modelo para geração automática de operadores morfológicos binários (capítulo 7), com base no Teorema da Decomposição Canônica da *MM* e em modelos de aprendizado computacional (capítulo 5).

Nesta modelagem, tivemos a preocupação de criar uma estrutura "formalmente completa" e útil do ponto de vista de um usuário pouco familiarizado com os resultados teóricos. A parte formal da estrutura está solidamente baseada nos resultados da *MM*, no modelo de aprendizado "PAC" e suas extensões, no fato da existência de isomorfismo entre o reticulado dos operadores morfológicos binários e o reticulado das funções Booleanas e na modelagem do domínio das imagens por "random sets". Os elementos formais garantem as qualidades "abrangência", "precisão" e "formalismo" ao modelo. Por outro lado, a forma pela qual o problema que se deseja resolver é especificada no modelo, garante-lhe a "facilidade de uso".

Dois pontos foram importantes para o sucesso deste trabalho. O primeiro foi a abordagem na qual inserimos o problema de programação automática dentro do contexto de aprendizado[1]. O segundo, foi o desenvolvimento de um algoritmo de aprendizado "eficiente". Esse algoritmo (ISI), que também pode ser utilizado para minimização de funções Booleanas a partir de sua forma canônica, não foi encontrado na literatura consultada e mostra-se muito melhor que outros algoritmos clássicos, quando a função Booleana possui muitos "don't cares".

Podemos dizer que este é um trabalho bastante completo, que envolveu estudos teóricos (Morfologia Matemática, Random Sets, Estimação Estatística, Algoritmos, PAC Learning), modelagem de um sistema, implementação (deste modelo como um sistema computacional - toolbox PAC), aplicações (utilização deste sistema para resolver problemas reais) e análise dos resultados. Os vários experimentos realizados permitiram verificar resultados altamente positivos e animadores (capítulo 9 e 10).

No capítulo 4 apresentamos os trabalhos realizados por Dougherty para estimação de filtros morfológicos ótimos. Embora nosso trabalho apresente alguns pontos em comum com o trabalho de Dougherty, destacamos que é mais geral pois :

- inserimos a estimação do operador dentro do contexto de aprendizado PAC;
- as aplicações não são restritas a problemas de restauração. Consideramos todos os operadores i.t. localmente definidos e variadas aplicações (capítulos 9 e 10);
- utilizamos o conceito de função de perda, para caracterizar o erro de um operador (Dougherty utiliza somente o MAE);
- utilizamos o algoritmo ISI para calcular a base, que é muito melhor que o algoritmo QM para diversos casos reais.

Um ponto delicado quando a escolha do operador ótimo é realizado a partir de amostras é que não sabemos determinar a sua "qualidade", pelo fato deste ter sido obtido a partir de uma distribuição de probabilidade estimada. O mérito da inserção do problema de estimação no contexto de aprendizado PAC é que ele possui bases teóricas sólidas para formalizar e tratar esta questão, ainda que os valores teóricos nele estabelecidos sejam diferentes daqueles que observamos na prática.

11.2 Futuras Considerações

Acreditamos que este trabalho vem reforçar ainda mais as evidências de que a *MM* é uma poderosa ferramenta para tratamento de imagens e que pode ser aplicada inclusive a problemas classicamente abordados por redes neurais, como reconhecimento de caracteres.

[1] Esta abordagem é inédita.

De fato, o nosso trabalho aponta a *MM* como uma alternativa para as redes neurais supervisionadas ([Has95]).

Certamente existem vários pontos citados neste trabalho que ainda merecem um estudo mais profundo. A seguir procuraremos analisar vários destes pontos e indicar algumas investigações que podem ser realizadas em relação a cada um deles.

Não existe ainda nenhum critério conhecido para escolha do tamanho ideal de janela. Também não existe nenhum estudo sobre a influência do tamanho da janela (considerada no treinamento) sobre a qualidade do resultado final.

Intuitivamente, parece-nos que este estudo relaciona-se diretamente com o estudo do domínio das imagens, ou seja, das características das imagens. Um recurso que possibilita a modelagem do domínio das imagens são os "random sets", que inserem as imagens e os operadores morfológicos no contexto estatístico.

Os experimentos indicam que a modelagem do domínio das imagens poderá também ser investigada para reduzir o espaço de busca dos operadores. Como o conhecimento sobre o domínio das imagens pode ser inserido analiticamente no contexto formal dos operadores ?

Nesta mesma linha de questionamento, certas propriedades dos operadores implicam a simplificação da sua decomposição (como o caso dos operadores crescentes). Quais outras propriedades acarretam simplificações na fórmula de decomposição dos operadores ? Será possível combinar duas propriedades e simplificar ainda mais a decomposição ? Se existirem respostas afirmativas a estas questões, provavelmente isto implicará uma redução no espaço de busca. Ainda no caso afirmativo, de que modo as eventuais formas simplificadas poderão ser inseridas no contexto de aprendizado ?

Em relação ao algoritmo ISI, utilizado para o aprendizado de funções Booleanas, um grande desafio consiste em investigar a sua complexidade de tempo e espaço. Uma outra questão que também merece ser investigada no tocante a este algoritmo e a de suas variantes é o comportamento dos mesmos em relação ao tratamento de "don't cares". Ou seja, de que maneira eles diferem em se tratando de atribuição de valores aos "don't cares" ? Este estudo pode ser importante para determinar quais das variantes devem ser utilizadas em um determinado processo de aprendizado. É melhor aquele que atribui valor 1 ao menor número possível de "don't cares" ? Ou é melhor aquele que atribui valor 1 aos "don't cares" que são "parecidos" com os exemplos positivos ?

Quanto ao processo de aprendizado, uma incógnita é o valor de m_0, isto é, o número de exemplos de treinamento necessários para garantir um "bom resultado". Verificamos nos experimentos que, os limites teóricos conhecidos para m_0 parecem ser muito superiores em relação aos valores experimentados na prática. Isto sugere a necessidade de uma maior investigação para determinar um limite mais justo.

Outra questão intrigante são os treinamentos de múltiplos estágios, que parece ser um campo fértil de pesquisa, com interessantes aplicações em problemas de classificação.

Finalizamos este texto com esta série de perguntas. Para responder estas questões, parece-nos que várias teorias tais como *MM*, teoria das probabilidades, "Random Sets", complexidade de algoritmos, teoria de aprendizado computacional, entre outros devem ser estudados com maior profundidade e conjugados entre si.

Bibliografia

[AB92] M. Anthony and N. Biggs. *Computational Learning Theory - An Introduction.* Cambridge University Press, 1992.

[Amm92] L. Ammeraal. *Programs and Data Structures in C.* John Wiley and Sons, second edition, 1992.

[Bar93] J. Barrera. *Uma abordagem unificada para problemas de Visão Computacional: a Morfologia Matemática.* PhD thesis, Departamento de Engenharia Eletrônica da EPUSP, março 1993.

[Bax94] G. A. Baxes. *Digital Image Processing - Principles and Applications.* John Wiley and Sons, Inc., 1994.

[BB91] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM. J. APPL. MATH.*, 51:1782–1792, December 1991.

[BB92] J. Barrera and G. J. F. Banon. Expressiveness of the Morphological Language. In *Image Algebra and Morphological Image Processing III*, volume 1769, pages 264–275, San Diego, CA, 1992. SPIE Proceedings.

[BB93] G. J. F. Banon and J. Barrera. Decomposition of Mappings Between Complete Lattices by Mathematical Morphology - Part I. General Lattices. *Signal Processing*, 30:299–327, 1993.

[BB94] G. J. F. Banon and J. Barrera. Bases da Morfologia Matemática para Análise de Imagens Binárias. IX Escola de Computação, Pernambuco, 1994.

[BBL94] J. Barrera, G. Banon, and R. Lotufo. A Mathematical Morphology Toolbox for the KHOROS System. In *Image Algebra and Morphological Image Processing V*. SPIE Proceedings, july 1994.

[Bir67] G. Birkhoff. *Lattice Theory.* American Mathematical Society, Providence, Rhode Island, 1967.

[Bre87] P. Bremaud. *An Introduction to Probabilistic Modeling.* Springer-Verlag, New York, 1987.

[BS95] J. Barrera and G. P. Salas. Set Operations on Collections of Closed Intervals and their Applications to the Automatic Programming of Morphological Machines. Technical Report RT-MAC-9413, Departamento de Ciência da Computação, IME-USP, Agosto 1995.

[BTdST95] J. Barrera, N. S. Tomita, F.S.C. da Silva, and R. Terada. Automatic Programming of Binary Morphological Machines by PAC Learning. In *Neural and Stochastic Methods in Image and Signal Processing*, volume 2568, pages 233–244, San Diego, 1995. SPIE Proceedings.

[BTdST96] J. Barrera, R. Terada, F.S.C. da Silva, and N. S. Tomita. Automatic Programming of MMachs for OCR. *a ser publicado no International Symposium on Mathematical Morphology*, 1996.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[DL94] E. R. Dougherty and R. P. Loce. Precision of Morphological-representation Estimators for Translation-invariant Binary Filters: Increasing and Nonincreasing. *Signal Processing*, 40:129–154, 1994.

[Dou92a] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters - I. Optimal Binary Filters. *CVGIP: Image Understanding*, 55(1):36–54, January 1992.

[Dou92b] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters - II. Optimal Gray Scale Filters. *CVGIP: Image Understanding*, 55(1):55–72, January 1992.

[Fil80] E. A. Filho. *Teoria Elementar dos Conjuntos.* Livraria Nobel S.A., São Paulo, 1980.

[GJ79] M. R. Garey and D. S. Jonhson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W.H. FREEMAN AND COMPANY, San Francisco, 1979.

[Gou92] J. Goutsias. Morphological Analysis of Discrete Randon Sets. *Journal of Mathematical Imaging and Vision*, 2:193–215, 1992.

[Grä78] G. Grätzer. *General Lattice Theory.* Academic Press, 1978.

[GW92] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[Had50] H. Hadwiger. Minkowskische Addition und Subtraktion beliebiger Punktmengen und die Theoreme von Erhard Schmidt. *Math. Zeitschrift*, 53:210–218, 1950.

[Had57] H. Hadwiger. *Vorlesungen über Inhalt, Oberfläche und Isoperimetrie.* Springer-Verlag, Berlin, 1957.

[Hal79] E. L. Hall. *Computer Image Processing and Recognition.* Academic Press, 1979.

[Has95] M. H. Hassoun. *Fundamentals of Artificial Neural Net.* MIT Press, 1995.

[Hau92] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and other Learning Applications. *Information and Computation*, 100:78–150, 1992.

[HP81] F. J. Hill and G. R. Peterson. *Introduction to Switching Theory and Logical Design.* John Wiley, 1981.

[HSZ87] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(4), July 1987.

[Ken74] D. G. Kendall. Foundations of a Theory of Random Sets. In E.F. Harding and D. G. Kendall, editors, *Stochastic Geometry*, pages 322–376. John Wiley & Sons, London, England, 1974.

[Knu73] D.E. Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley, 1973.

[KS90] M. J. Kearns and R. E. Schapire. Efficient Distribution-free Learning of Probabilistic Concepts. In *31st Annual IEEE Symposium on Foundations of Computer Science*, volume 1, pages 382–391, 1990.

[Loc93] R. P. Loce. *Morphological Filter Mean-Absolute-Error Representation Theorems and Their Application to Optimal Morphological Filter Design.* PhD thesis, Center of Image Processing - Rochester Institute of Technology, 1993.

[Mar85] P. A. Maragos. *A Unified Theory of Translation-invariant Systems with Applications to Morphological Analysis and Coding of Images.* PhD thesis, School of Elect. Eng. - Georgia Inst. Tech., 1985.

[Mat75] G. Matheron. *Random Sets and Integral Geometry.* John Wiley, 1975.

[Mey69] P. L. Meyer. *Probabilidade - Aplicações à Estatística.* Ao Livro Técnico S.A., Rio de Janeiro, 1969.

[Min03] H. Minkowski. Volumen und Oberfläche. *Math. Ann.*, 57:447–495, 1903.

[MSM63] G. D. Mostow, J. H. Sampson, and J. Meyer. *Fundamental Structures of Algebra.* McGraw-Hill, 1963.

[Pra91] W. K. Pratt. *Digital Image Processing.* John Wiley and Sons, Inc., 1991.

[RASW90] J. Rasure, D. Argiro, T. Sauer, and C. Williams. Visual Language and Software Development Environment for Image Processing. *International Journal of Imaging Systems and Technology*, 2:183–199, 1990.

[Ros69] A. Rosenfeld. *Picture Processing by Computer.* Academic Press, 1969.

[Rut65] D. E. Rutherford. *Introduction to Lattice Theory.* Oliver and Boyd LTD, London, 1965.

[Sch89] M. Schimitt. *Des Algorithmes Morphologiques à l'intelligence Artificielle.* PhD thesis, École Nationale Superiéure des Mines de Paris, Fontainebleau, 1989.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[Ser88] J. Serra. *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances.* Academic Press, 1988.

[Szá63] G. Szász. *Introduction to Lattice Theory.* Academic Press, 1963.

[TG74] J.T. Tou and R. C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley, 1974.

[TLR90] T.H.Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[Val84] L. Valiant. A Theory of the Learnable. *Comm. ACM*, 27:1134–1142, 1984.