**Fabrício Enembreck**

# Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas

Dissertação apresentada à Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Ciências

**Curitiba**

**1999**

**Fabrício Enembreck**

# Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas

Dissertação apresentada à Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Ciências

Área de Concentração:
Sistemas Inteligentes

Orientador:
Prof. Dr. Bráulio Coelho Ávila

Co-orientador:
Prof. Dr. Robert Sabourin

**Curitiba**

**1999**

Enembreck, Fabrício

Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas. Curitiba, 1999.
132p.

Dissertação (Mestrado) - Pontifícia Universidade Católica do Paraná. Departamento de Informática.

1. Verificação de Assinaturas 2. Sistemas Inteligentes I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Departamento de Informática II. t

*Aos meus pais José e Carmela,*
*pelo sacrifício realizado em meu benefício*
*e pelo constante apoio e dedicação.*

*Aos meus tios Antônio Carlos e Dirce,*
*pelo apoio, incentivo e amizade a mim demonstrada.*

*Às minhas sobrinhas Fábia e Flávia,*
*pela alegria e carinho que representam em minha vida.*

# Agradecimentos

Ao Prof. Bráulio Coelho Ávila pelas várias horas de trabalho dedicadas à realização deste trabalho;

Ao Prof. Robert Sabourin, pelas idéias e comentários que me ajudaram a alcançar os resultados apresentados neste trabalho;

Ao Prof. Alex Alves Freitas, pelos ricos comentários e sugestões relacionados aos temas de pesquisa abordados;

Às amigas Elaini, Katiana e Regiane pelos momentos difíceis por nós superados durante a realização das disciplinas;

A Luis Torgo e Thomas Hoppe, pela disponibilização de alguns algoritmos utilizados neste trabalho;

À PUC-PR pela oportunidade concedida e pelo apoio financeiro;

A todos aqueles que me ajudaram direta ou indiretamente na realização deste trabalho.

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas

AD. . . . . .Árvore de Decisão
ADP. . . . . Árvore de Decisão Paraconsistente
AM. . . . . Aprendizado de Máquina
DM. . . . . .Data Mining
DST. . . . . .Teoria de Dempster-Shapher
FC. . . . . .Fator de Certeza
GI. . . . . Gerenciamento de Incerteza
IA. . . . . Inteligência Artificial
I/S . . . . . .Inconsistência e Subdeterminação
LP. . . . . .Lógica Paraconsistente
MC. . . . . Medida de Crença
MD. . . . . Medida de Descrença
MMD. . . . . Mínima Distância
NN. . . . . Vizinho mais Próximo
PLE. . . . . .Programa Lógico Evidencial
PrLE. . . . . Programação Lógica Evidencial
TP. . . . . Teoria da Probabilidade

# Capítulo 1

# Introdução

Pesquisadores de Inteligência Artificial [Russel, 1995] [Rich, 1994] [Shohan, 1994] [Luger, 1993] — IA — trabalham continuamente no desenvolvimento de sistemas capazes de realizar tarefas que, geralmente, são realizadas por seres humanos. Por exemplo, jogar xadrez, diagnosticar doenças, etc. Porém, atividades inerentes e triviais para seres humanos como pensar, reconhecer, aprender, etc., representam grandes desafios à IA. Essas tarefas tornam-se mais desafiadoras à medida que podem ser realizadas, de maneira natural, por qualquer pessoa, e, às vezes, executadas de maneira extremamente limitada, pelos mais poderosos computadores, desenvolvidos através de anos de pesquisas.

Dentre as tarefas mencionadas, o aprendizado constitui uma das tarefas mais estudadas atualmente. O aprendizado computadorizado — Aprendizado de Máquina, AM — utiliza alguns conceitos envolvidos no aprendizado humano para adquirir conhecimento. Assim como uma criança necessita de diversas observações de um objeto para aprender a reconhecê-lo, um computador também necessita de inúmeras observações — tuplas de uma base de dados — para adquirir algum conhecimento, geralmente, representado na forma de regras [Fu, 1996] [Hätönen, 1996] [Jim, 1995].

Dentro de AM, diversos paradigmas foram desenvolvidos. Um dos paradigmas

mais conhecidos e estudados é o paradigma simbólico através de Árvore de Decisão — AD. Como descrito anteriormente, o conhecimento computadorizado é adquirido através da aplicação de algoritmos de AM sobre bases de dados, gerando conceitos que são representados de alguma forma. Porém, uma série de problemas pode surgir durante este processo, como por exemplo, imperfeições nos dados, eficiência do algoritmo, performance de processamento, etc.

A ocorrência de dados conflitantes[1] — inconsistência — em bases de dados afeta negativamente a performance dos algoritmos de AM. Geralmente, a existência de informações conflitantes está relacionada à união de bases de dados geradas a partir de fontes distribuídas, à erro na coleta dos dados e à falta de variáveis necessárias para aquisição do conhecimento.

Em IA, geralmente, técnicas de Gerenciamento de Incerteza — GI — são empregadas na construção de modelos de inferência sobre informações imperfeitas, como por exemplo, Teoria da Probabilidade, Teoria de Dempster-Shafer, Fatores de Certeza e Lógica Fuzzy. Porém, nenhum desses formalismos foi desenvolvido especificamente para manipulação de informações inconsistentes.

Em [da Costa, 1963] [da Costa, 1974] foram apresentados conceitos de *Lógica Paraconsistente* — LP. Tal formalismo teve como motivação principal, permitir interpretações adequadas para situações onde são encontradas inconsistências. Dessa forma, inconsistências não são eliminadas no processo de inferência. Além disso, outras imperfeições como incerteza e ignorância, também podem ser tratadas através da LP. A Lógica Paraconsistente — LP — representa uma extensão à Lógica Clássica, e permite que informações contraditórias — $p$ e $\neg p$ — estejam presentes ao mesmo tempo em um sistema.

Posteriormente ao desenvolvimento da LP, foi desenvolvida a *Programação Lógica*

[1]Neste trabalho, entende-se por dados conflitantes, ou inconsistentes, bases de dados que apresentam exemplos cujos valores dos atributos previsores são idênticos, porém, diferem quanto ao valor do atributo meta.

*Evidencial* — PrLE — [Blair, 1987] [Blair, 1988] [Kifer, 1989] [Subrahmanian, 1987a] [Subrahmanian, 1987b] [Ávila, 1997]. Em PrLE, são fornecidos mecanismos que permitem quantificar a inconsistência de uma determinada inferência, realizada sobre um conjunto de informações. Em cada item de informação de um *Programa Lógico Evidencial* — PLE —, estão associadas evidências que são combinadas para a obtenção de um grau de inconsistência geral para a inferência.

A utilização de conceitos de PrLE pode contribuir no desenvolvimento e adaptação de algoritmos de AM capazes de adquirir conhecimento útil, mesmo quando informações contraditórias estão presentes em bases de dados. Tais algoritmos podem ser de extrema utilidade, uma vez que bases de dados inconsistentes são comuns em quaisquer domínios de aplicação.

Um dos objetivos deste trabalho é estender o algoritmo básico de AD, de forma a empregar conceitos de PrLE, gerando uma *Árvore de Decisão Paraconsistente* — ADP. Exemplos de bases de dados são adaptados de forma a apresentarem evidências. Estas evidências são calculadas automaticamente segundo conceitos da Teoria dos Fatores de Certeza. O modelo de inferência da PrLE é utilizado de forma a permitir ao sistema inferir confiavelmente mesmo quando informações conflitantes estão presentes na base de dados.

O modelo de AM implementado pode tanto realizar a classificação de um exemplo, cuja classe seja desconhecida, quanto fornecer fatores evidenciais — *Crença* e *Descrença* — sobre a inconsistência de determinado exemplo, cuja classe seja conhecida. No último caso, as evidências podem ser combinadas de forma a se obter o grau de inconsistência do exemplo em relação ao modelo representado pela AD. Essa característica, torna possível a sua aplicação na tarefa de verificação de assinaturas manuscritas.

Em um ambiente bancário, milhares de cheques são verificados manualmente todos os dias e diversos problemas estão relacionados à esta tarefa — incapacidade

técnica da pessoa responsável pelo reconhecimento dos campos dos cheques, tempo envolvido para a tarefa, idoneidade do responsável, etc. Dessa forma, o reconhecimento automatizado de assinaturas manuscritas [Sabourin, 1993] [Sabourin, 1994] [Bastos, 1995] torna-se uma tarefa extremamente útil, pois poderia minimizar os problemas relacionados à autenticação de documentos. Além do ambiente bancário, o reconhecimento automatizado de assinaturas manuscritas possui inúmeras oportunidades de aplicação. Isso porque documentos que realizam identificação pessoal geralmente utilizam assinaturas como forma de autenticação.

Em [Sabourin, 1997] foi desenvolvido um formalismo para verificação de assinaturas — sistema RECSIG. Tal sistema implementa uma forma de representação local de imagens de assinaturas. Ou seja, cada imagem é dividida em subregiões chamadas retinas, e características são extraídas a partir dos traços presentes em cada retina. Segundo Sabourin, características locais referentes à subregiões de imagens de assinaturas manuscritas são muito mais discriminantes que características baseadas no formato geral da assinatura. Dessa forma, a representação de uma assinatura é constituída por um conjunto de exemplos, referentes às retinas.

Um outro objetivo do trabalho consiste em utilizar o modelo de AM desenvolvido no problema de verificação de assinaturas. O sistema desenvolvido é aplicado sobre as bases de dados obtidas a partir da base de imagens de assinaturas descrita em [Sabourin, 1997]. Através da ADP, dada uma assinatura de entrada, pode-se obter fatores evidenciais referentes à subregiões da assinatura. Posteriormente, esses fatores podem ser combinados de forma que um valor total de inconsistência possa ser atribuído à assinatura em relação ao modelo representado pela ADP.

Neste trabalho procura-se também avaliar o comportamento do sistema comparando-o com um algoritmo, baseado em AD muito conhecido da comunidade de AM — C4.5. Ambos os sistemas — ADP e C4.5 — são aplicados sobre diversas bases de dados de diversos domínios. Os algoritmos são comparados em termos de

precisão de classificação e tamanho das árvores geradas. Com o objetivo de apresentar o comportamento de ambos os algoritmos quando aplicados sobre bases de dados inconsistentes, tais bases posteriormente foram modificadas, de forma a apresentarem exemplos inconsistentes. Dessa forma, pode-se constatar a utilidade e as contribuições do sistema desenvolvido.

O presente trabalho está organizado da seguinte forma: Capítulo 2 apresenta os conceitos envolvidos em AM e AD; posteriormente, técnicas de GI são descritas no Capítulo 3. No Capítulo 4, são abordados os conceitos de inferência sobre informações inconsistentes através da PrLE. Processamento de Imagens e técnicas empregadas no sistema de reconhecimento de assinaturas manuscritas RECSIG são descritas no Capítulo 5. O modelo de inferência proposto em Árvore de Decisão Paraconsistente e a comparação com o algoritmo C4.5 são apresentados no Capítulo 6. A aplicação do sistema no problema de verificação de assinaturas e os resultados obtidos são descritos no Capítulo 7. As conclusões finais, os resultados obtidos com a pesquisa realizada, possíveis extensões e trabalhos futuros são apresentados no Capítulo 8.

# Capítulo 2

# Aprendizado de Máquina

## 2.1 Considerações Iniciais

A habilidade de aprender novos conceitos, inerente a todo ser humano, constitui um grande desafio para pesquisadores de IA. Assim como um estudante é capaz de aprender sobre as mais variadas áreas, pesquisadores de IA, mais especificamente de AM, trabalham no desenvolvimento de sistemas computacionais capazes de adquirir conhecimento e utilizá-lo para desempenhar um comportamento inteligente em domínios genéricos de aplicação. Por exemplo, a tarefa de identificação do autor de uma assinatura manuscrita pode ser facilmente realizada por um ser humano com uma pequena taxa de erro. *Como isso é possível?* O mecanismo completo de aprendizado humano não é totalmente conhecido. Porém, sabe-se que a versão computadorizada de alguns elementos que constituem o mecanismo de aprendizado humano, tais como visão [Murtagh, 1996] [Murtagh, 1996a] [Weeks, 1996], memória e raciocínio, ou ainda não estão disponíveis, ou se apresentam de maneira extremamente simplificada, como é o caso da visão.

O desconhecimento sobre a forma exata do aprendizado humano estimulou pesquisadores de AM a desenvolver diversos paradigmas de aprendizado computacional

— Paradigma Simbólico [Monard, 1997] [Quinlan, 1986] [Quinlan, 1993] [Clark, 1989] [Clark, 1991] [Pitas, 1992], Estatístico [Schalkoff, 1992] [Mitchell, 1997], Paradigma Baseado em Instância [Rich, 1994] [Russel, 1995] [Nadler, 1993] [Schalkoff, 1992], Paradigma Genético [Tanomaru, 1995] e Paradigma Coneccionista [Schalkoff, 1992] [Gall, 1992].

Neste trabalho, será dado ênfase ao paradigma Simbólico de AM, mais especificamente, aprendizado através de Árvores de Decisão — AD [Quinlan, 1993] [Evans, 1994] [Lovell, 1996] [Kamber, 1998] [Janikow, 1998]. Através deste paradigma, conceitos podem ser gerados a partir de observações e estruturados em forma de AD, tal que cada vértice da árvore representa um determinado atributo e um subconjunto de exemplos e ligações entre os vértices representam testes sobre estes atributos.

Atualmente, grande parte das aplicações que envolvem AM e Data Mining — DM — [Jim, 1995] [Wüthrich, 1997] [Holsheimer, 1994] [Fayad, 1996] [Ávila, 1998] [Sasisekharan, 1996] [Hätönen, 1996] [Freitas, 1998] apresenta módulos que implementam o processo de indução [Clark, 1993] a partir de AD. Este grande interesse científico e comercial sobre algoritmos de AD se deve, principalmente, à alta compreensibilidade do conhecimento extraído e generalizado a partir dos dados, à simplicidade de implementação de tais algoritmos e à qualidade dos conceitos gerados em termos de classificação, geralmente satisfatória para a maioria das aplicações.

No Apêndice A.2 são apresentados os principais conceitos envolvidos na tarefa de AM. Nesta Seção um método de AM em particular será detalhado: AD.

## 2.2 Aprendizado por Árvores de Decisão

Árvores de Decisão constituem um dos formalismos mais conhecidos de AM. A partir do desenvolvimento do sistema ID3 [Quinlan, 1986], algoritmos da família

TDIDT — *Top Down Induction of Decision Tree* — tornaram-se uma referência para a maioria dos trabalhos que utilizam aprendizado supervisionado a partir de exemplos em sistemas de IA.

Idealizado na década de 50 por Hunt [Hunt, 1966], o algoritmo constrói uma AD através de sucessivos particionamentos, dirigidos pela freqüência de informação. Desta forma, a ordem dos exemplos não influencia o processo, porém, o aprendizado de AD é não-incremental[1].

O objetivo é construir AD que representem os padrões implícitos nos dados e que não podem, pelo menos facilmente, ser descobertos a partir de processo manual, uma vez que o número de observações e a quantidade de atributos é geralmente alta. Outra característica dos algoritmos da família TDIDT é a geração de árvores simples, porém, isso não garante que a melhor árvore seja obtida, pois árvores menores, geralmente, apresentam menor precisão na classificação de exemplos não conhecidos [Murphy, 1994].

### 2.2.1 O Algoritmo de Árvores de Decisão

O algoritmo básico de construção de AD é simples. Considerando um conjunto $E = \{e_1, ..., e_n\}$ de exemplos de treinamento e um conjunto $C = \{c_1, ..., c_m\}$ de classes mutuamente exclusivas que podem ser associadas a um exemplo $e_i \in E$, o algoritmo básico de AD é descrito a seguir:

1. Se $E$ é um conjunto não-vazio de exemplos pertencentes à mesma classe $c_j$, então formar uma folha a partir de $E$ e $c_j$ e encerrar;

[1] Um sistema que realiza aprendizado incremental é capaz de obter conhecimento à medida que novas observações são dinamicamente fornecidas ao sistema. Dessa forma, quando uma observação inconsistente em relação aos conceitos gerados é fornecida, o sistema modifica apenas os conceitos inconsistentes de forma a classificar esta nova observação corretamente e continua o processamento sem alterar a parte restante do conhecimento.

2. Se $E$ é um conjunto vazio — não possui exemplos —, então formar uma folha e escolher, segundo algum critério, a classe $c_i$ para a folha. O algoritmo C4.5 [Quinlan, 1993] utiliza a classe mais freqüente existente no nó-pai;

3. Se $E$ possui exemplos que pertencem a diferentes classes, então escolher um atributo $a_i \in \{a_1, ..., a_o\}$, particionar o conjunto $E$ em subconjuntos $\{E_1, ..., E_k\}$, de acordo com os $k$ valores possíveis para o atributo $a_i$ — domínio do atributo $a_i$ — e realizar o mesmo processo recursivamente, retornando ao item 1, para cada $E_j \in \{E_1, ..., E_k\}$.

Um exemplo clássico (Exemplo 2.2.1), que ilustra o processo de criação de uma AD [Mitchell, 1997] [Rich, 1994], consiste de um pequeno conjunto de treinamento que possui 14 exemplos, 4 atributos previsores e 2 classes mutuamente exclusivas. Os exemplos representam observações metereológicas relacionadas à prática e não-prática de esporte.

**Exemplo 2.2.1** A Tabela 2.1 representa um conjunto de observações que relaciona dados climáticos e a prática e não-prática de esportes.

Quando o algoritmo é iniciado, tem-se $E = \{e_1, ..., e_{14}\}$. Verifica-se que a terceira cláusula do algoritmo é satisfeita, portanto, um dos atributos precisará ser escolhido. Assumindo arbitrariamente a escolha do atributo *Exterior*, o conjunto $E$ seria subdividido como ilustrado na Figura 2.1.

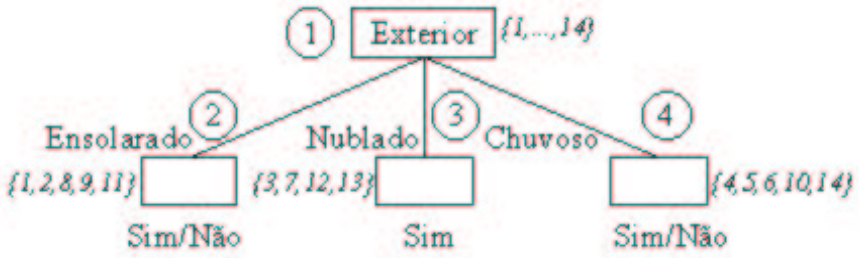



Figura 2.1: Subdivisão do Conjunto pelo Atributo Exterior.

| $_i$ | Exterior | Temperatura | Umidade | Vento | Prática de Esporte |
|---|---|---|---|---|---|
| 1 | ensolarado | alta | alta | não | não |
| 2 | ensolarado | alta | alta | sim | não |
| 3 | nublado | alta | alta | não | sim |
| 4 | chuvoso | média | alta | não | sim |
| 5 | chuvoso | baixa | normal | não | sim |
| 6 | chuvoso | baixa | normal | sim | não |
| 7 | nublado | baixa | normal | sim | sim |
| 8 | ensolarado | média | alta | não | não |
| 9 | ensolarado | baixa | normal | não | sim |
| 10 | chuvoso | média | normal | não | sim |
| 11 | ensolarado | média | normal | sim | sim |
| 12 | nublado | média | alta | sim | sim |
| 13 | nublado | alta | normal | não | sim |
| 14 | chuvoso | média | alta | sim | não |

Tabela 2.1: Exemplo de uma Base de Dados sobre Prática de Esporte.

Prosseguindo a execução do algoritmo, verifica-se que todos os exemplos presentes no nó 3 da árvore pertencem à mesma classe, portanto, a primeira sentença do algoritmo é executada e uma folha é gerada. Por outro lado, para os nós 2 e 4, o algoritmo necessita continuar.

Uma árvore que potencialmente poderia ser gerada para o exemplo é a ilustrada na Figura 2.2.

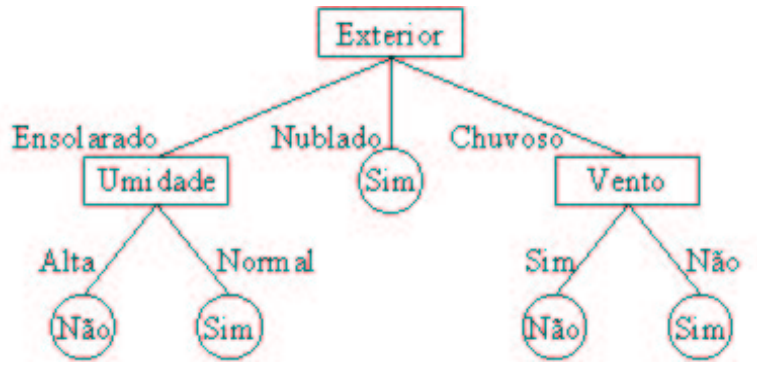



Figura 2.2: Árvore de Decisão Gerada com Poucos Nós.

A classificação de um exemplo é obtida percorrendo a árvore da raiz até uma determinada folha, realizando, para isso, os devidos testes. A folha encontrada representa a classe do exemplo.

Os conceitos obtidos pela AD apresentada anteriormente podem ser facilmente traduzidos no formato de regras. A seguir são apresentados os conceitos equivalentes à AD anterior.

**se** $Exterior = ensolarado$ **e** $Umidade = alta$ **então** $classe = não$

**se** $Exterior = ensolarado$ **e** $Umidade = normal$ **então** $classe = sim$

**se** $Exterior = nublado$ **então** $classe = sim$

**se** $Exterior = chuvoso$ **e** $vento = sim$ **então** $classe = não$

**se** $Exterior = chuvoso$ **e** $vento = não$ **então** $classe = sim$

### 2.2.2 Escolha do Atributo

No exemplo fornecido na Seção anterior, a escolha do atributo durante a geração da árvore foi arbitrária. Porém, esta etapa é fundamental para o algoritmo, e influencia diretamente o tamanho, e, portanto, a qualidade da árvore. Logo, o método de escolha necessita ser descrito detalhadamente.

Para demonstrar a importância do critério de escolha, a Figura 2.3 mostra uma outra AD que poderia ser gerada, a partir do mesmo conjunto de treinamento fornecido na Seção anterior, utilizando-se um critério de escolha qualquer.

Obviamente, a complexidade da árvore mostrada na Figura 2.3 é muito maior em relação à árvore da Figura 2.2. Estudos realizados por Murphy [Murphy, 1994], demonstram que, na média, grandes ADs tendem a apresentar uma maior precisão da classificação de exemplos não conhecidos. Porém, na prática, ADs menores são preferíveis porque geram conceitos que são mais facilmente compreendidos por hu-

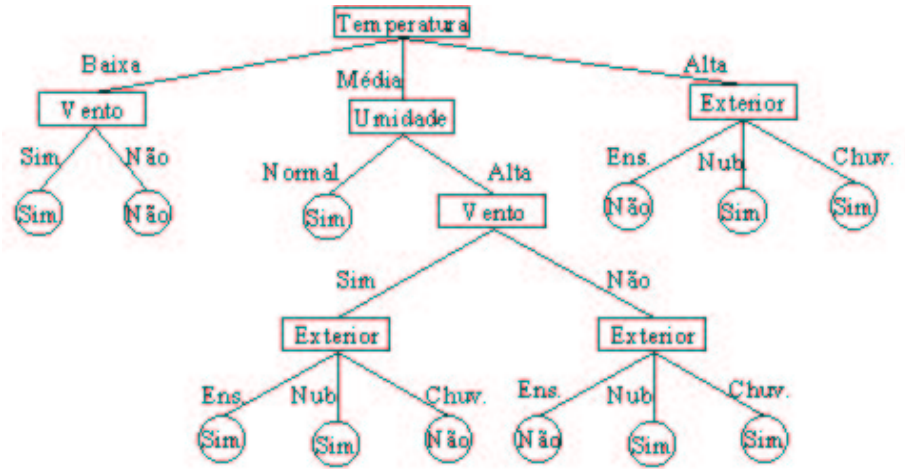



Figura 2.3: Árvore de Decisão Complexa.

manos. Essa característica, geralmente, tende a ser mais utilizada em aplicações onde o conhecimento extraído é mais valioso que a própria classificação e o nível de abstração necessário é relativamente alto. Partindo deste princípio, algoritmos da família TDIDT tendem a encontrar árvores simplificadas que representam o modelo dos dados através de uma escolha determinística dos atributos.

### 2.2.3 Critério de Escolha

As únicas informações disponíveis para utilização do critério de escolha de um determinado atributo $a_i$ são o conjunto de atributos $A = \{a_1, ..., a_m\}$ e a distribuição das classes $C = \{c_1, ..., c_k\}$ existentes no conjunto de treinamento $E = \{e_1, ..., e_n\}$.

Hunt [Hunt, 1966] sugeriu que critérios de escolha baseados na teoria da informação [Epstein, 1988] [Mackay, 1997] deveriam obter resultados mais significativos do que os inicialmente utilizados, que utilizavam apenas freqüência de classes [Quinlan, 1993].

Quinlan [Quinlan, 1986] utilizou o critério de escolha baseado no ganho de informação. Esse critério utiliza como base o conceito da teoria da informação de que a informação contida em uma determinada mensagem depende de sua proba-

bilidade e pode ser medida em bits através do logaritmo negativo na base 2 dessa probabilidade.

Pode-se calcular, por exemplo, a probabilidade de um determinado exemplo estar associado a uma classe $c_j \in C$ da seguinte forma:

$$p_j = \frac{|E_{c_j}|}{|E|} \tag{2.1}$$

tal que, $E_{c_j}$ é o subconjunto de $E$ formado pelos exemplos pertencentes à classe $c_j$. Portanto, a quantidade de informação contida em tal probabilidade é:

$$inf(p_j) = -log_2 \frac{|E_{cj}|}{|E|} \ bits \tag{2.2}$$

Para medir a impureza, ou desordem, de um conjunto $E$ de exemplos de treinamento é introduzida a medida de entropia dada por:

$$Entropia(E) = -\sum_{i=1}^{k} p_i \log_2 p_i \tag{2.3}$$

Tal medida pode ser interpretada como o número de bits de informação necessário para codificar a classificação de um exemplo pertencente a $E$. Assumindo que existem $k$ classes possíveis em $E$, a medida de informação necessária para classificar um exemplo corretamente é dada pelo somatório negativo da quantidade proporcional de bits associada à distribuição de todas as classes.

Por exemplo, a entropia do conjunto de treinamento $E$ apresentado no Exemplo 2.2.1 é dada por:

$$\begin{aligned} Entropia(E) &= -9/14 \times \log_2 9/14 - 5/14 \times \log_2 5/14 \\ Entropia(E) &= 0.940 \end{aligned}$$

Considerando um conjunto de exemplos $E$ e apenas duas classes $c_1$ e $c_2$, mutuamente exclusivas, a entropia de $E$ é 0 quando todos os exemplos pertencem à mesma classe $c_j$ — $|E| = |E_{c_j}|$ —, pois:

$$\begin{aligned} Entropia(E) &= -\frac{|E|}{|E|} \times \log_2 \frac{|E|}{|E|} \\ Entropia(E) &= 0 \end{aligned}$$

De forma semelhante, pode-se perceber que a entropia de $E$ será 1 quando o número de exemplos de ambas as classes for igual — $|E_{c_1}| = |E_{c_2}| = \frac{|E|}{2}$ —, pois

$$\begin{aligned} Entropia(E) &= -\frac{\frac{|E|}{2}}{|E|} \times \log_2 \frac{\frac{|E|}{2}}{|E|} - \frac{\frac{|E|}{2}}{|E|} \times \log_2 \frac{\frac{|E|}{2}}{|E|} \\ Entropia(E) &= -\frac{1}{2} \times \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1 \end{aligned}$$

Portanto, quando a distribuição de $c_1$ e $c_2$ é diferente, a entropia de $E$ estará entre 0 e 1 [Mitchell, 1997].

Uma outra medida necessária para o cálculo do ganho é a medida de informação obtida com o particionamento do conjunto $E$ em $\{E_1, ..., E_m\}$, de acordo com os $m$ valores pertencentes ao domínio do atributo escolhido $a$:

$$info(E, a) = \sum_{v \ \in \ dom(a)} \frac{|Ea_v|}{|E|} \times Entropia(Ea_v) \tag{2.4}$$

tal que, $dom(a)$ é o conjunto de valores possíveis para o atributo $a$ e $Ea_v$ é o subconjunto de $E$ formado por exemplos que possuem valor $v$ para o atributo $a$.

Finalmente, o ganho de informação obtido a partir de um conjunto de exemplos $E$ e um atributo $a$ é dado por:

$$ganho(E, a) = Entropia(E) - info(E, a) \tag{2.5}$$

Dessa forma, para cada atributo $a \in \{a_1, ..., a_m\}$ é calculado o ganho de informação e o atributo escolhido é aquele que maximiza o ganho.

Note-se que o segundo termo da equação do ganho é simplesmente o somatório da entropia calculada de cada subconjunto, multiplicado pela proporção que o subconjunto representa no conjunto total $E$. *Ganho* é, portanto, a redução esperada na entropia causada por um atributo $a$, ou ainda, a quantidade de informação obtida pelo particionamento de $E$ em $\{E_1, ..., E_m\}$, de acordo com os $m$ valores do atributo $a$.

Retornando ao Exemplo 2.2.1, os cálculos necessários para a escolha do atributo inicial são os seguintes:

$$
\begin{aligned}
Entropia(E) &= 0.940 \quad (calculado\ anteriormente) \\
info(E, exterior) &= \frac{5}{14} \times \left[-\frac{3}{5} \times \log_2 \frac{3}{5} - \frac{2}{5} \times \log_2 \frac{2}{5}\right] + \\
&\quad \frac{4}{14} \times \left[-\frac{4}{4} \times \log_2 \frac{4}{4}\right] + \\
&\quad \frac{5}{14} \times \left[-\frac{3}{5} \times \log_2 \frac{3}{5} - \frac{2}{5} \times \log_2 \frac{2}{5}\right] = 0.694 \\
info(E, Temperatura) &= \frac{4}{14} \times \left[-\frac{2}{4} \times \log_2 \frac{2}{4} - \frac{2}{4} \times \log_2 \frac{2}{4}\right] + \\
&\quad \frac{5}{14} \times \left[-\frac{4}{5} \times \log_2 \frac{4}{5} - \frac{1}{5} \times \log_2 \frac{1}{5}\right] + \\
&\quad \frac{5}{14} \times \left[-\frac{3}{5} \times \log_2 \frac{3}{5} - \frac{2}{5} \times \log_2 \frac{2}{5}\right] = 0.890 \\
info(E, Umidade) &= \frac{7}{14} \times \left[-\frac{4}{7} \times \log_2 \frac{4}{7} - \frac{3}{7} \times \log_2 \frac{3}{7}\right] + \\
&\quad \frac{7}{14} \times \left[-\frac{6}{7} \times \log_2 \frac{6}{7} - \frac{1}{7} \times \log_2 \frac{1}{7}\right] = 0.788 \\
info(E, Vento) &= \frac{8}{14} \times \left[-\frac{2}{8} \times \log_2 \frac{2}{8} - \frac{6}{8} \times \log_2 \frac{6}{8}\right] + \\
&\quad \frac{6}{14} \times \left[-\frac{3}{6} \times \log_2 \frac{3}{6} - \frac{3}{6} \times \log_2 \frac{3}{6}\right] = 0.892 \\
ganho(E, Exterior) &= 0.940 - 0.694 = 0.246 \quad \longleftarrow\ Maior\ Ganho \\
ganho(E, Temperatura) &= 0.940 - 0.890 = 0.05
\end{aligned}
$$

$$
\begin{aligned}
ganho(E, Umidade) &= 0.940 - 0.788 = 0.152 \\
ganho(E, Vento) &= 0.940 - 0.892 = 0.048
\end{aligned}
$$

Resultados interessantes podem ser encontrados através de ADs construídas utilizando-se o critério do ganho. Porém, o ganho apresenta uma séria deficiência, pois tende a favorecer a escolha de atributos que possuem muitos valores possíveis, ou seja, grande domínio. Esse problema pode ser observado, por exemplo, quando um atributo $a$ possui um valor associado a cada exemplo, logo, o valor de $info(E, a)$ será igual a 0 e seu ganho, portanto, é máximo [Quinlan, 1993]. Para minimizar o problema, Quinlan adotou uma medida de normalização calculada da seguinte maneira:

$$
info\_div(E, a) = - \sum_{v \in dom(a)} \left( \frac{|Ea_v|}{|E|} \right) \times \log_2 \left( \frac{|Ea_v|}{|E|} \right) \tag{2.6}
$$

Esta medida representa a informação potencial gerada pela divisão de $E$ em subconjuntos, enquanto a medida de informação relevante para a classificação é dada por:

$$
ganho\_médio(E, a) = \frac{ganho(E, a)}{info\_div(E, a)} \tag{2.7}
$$

O cálculo do *ganho_médio* para o Exemplo 2.2.1 é mostrado a seguir:

$$
\begin{aligned}
info\_div(E, Exterior) &= -\frac{5}{14} \times \log_2 \frac{5}{14} - \frac{4}{14} \times \log_2 \frac{4}{14} - \\
&\quad \frac{5}{14} \times \log_2 \frac{5}{14} = 1.577 \\
info\_div(E, Temperatura) &= -\frac{4}{14} \times \log_2 \frac{4}{14} - \frac{5}{14} \times \log_2 \frac{5}{14} - \\
&\quad \frac{5}{14} \times \log_2 \frac{5}{14} = 1.577 \\
info\_div(E, Umidade) &= -\frac{7}{14} \times \log_2 \frac{7}{14} - \frac{7}{14} \times \log_2 \frac{7}{14} = 1
\end{aligned}
$$

$$
\begin{aligned}
info\_div(E, Vento) &= -\frac{8}{14} \times \log_2 \frac{8}{14} - \frac{6}{14} \times \log_2 \frac{6}{14} = 0.985 \\
ganho\_médio(E, Exterior) &= \frac{0.246}{1.577} = 0.156 \\
ganho\_médio(E, Temperatura) &= \frac{0.05}{1.577} = 0.031 \\
ganho\_médio(E, Umidade) &= \frac{0.152}{1} = 0.152 \\
ganho\_médio(E, Vento) &= \frac{0.048}{0.985} = 0.048
\end{aligned}
$$

Muitos trabalhos foram desenvolvidos exclusivamente sobre escolha e relevância de atributos. Em [Kamber, 1998] é apresentada uma técnica de atribuição de um coeficiente de incerteza que pode eliminar atributos irrelevantes. O trabalho de [Kononenko, 1995] apresenta onze critérios de escolha de atributos e alguns resultados comparativos.

## 2.3 Características do Aprendizado de Máquina Baseado em AD

Não é difícil encontrar bases de dados onde tuplas possuem valores desconhecidos para determinados atributos. Várias técnicas foram desenvolvidas para tratamento de tal anomalia. Alguns autores abordam o problema atribuindo o valor *desconhecido*, um valor extra adicionado ao domínio de cada atributo, a valores faltantes de atributos. Outros trabalhos apresentam formas de estimar o valor desconhecido do atributo, por exemplo, considerando o valor mais freqüente. Alguns trabalhos, ainda, apenas excluem tuplas que possuem algum valor deconhecido e as consideram inválidas.

Em AD, duas decisões devem ser tomadas em relação a atributos com valores desconhecidos:

- como considerar valores desconhecidos durante o particionamento do conjunto;
- como classificar um exemplo com estas características.

No algoritmo C4.5, Quinlan utiliza a freqüência de valores conhecidos para o atributo com o objetivo de influenciar o ganho de informação. Dessa forma, o ganho é calculado da seguinte maneira:

$$ganho\_faltantes(E, a) = Freq(E, a) \times ganho\_médio(E, A) \tag{2.8}$$

tal que, $Freq(E, a)$ é a probabilidade do atributo apresentar valores conhecidos. Esta medida pode ser calculada a partir de:

$$Freq(E, a) = \frac{conhecidos(a)}{|E|} \tag{2.9}$$

tal que, $conhecidos(a)$ é o número de tuplas que possuem valores conhecidos para o atributo $a$.

Durante a classificação de um exemplo com valores desconhecidos, o sistema C4.5 considera todos os testes alternativos para o atributo e, a partir das folhas encontradas, retorna a classe com maior probabilidade de ocorrência.

Os exemplos apresentados até agora sobre aprendizado de AD utilizavam apenas atributos categóricos, que possuem poucos valores associados. Porém, é necessário tratar de maneira específica a divisão do conjunto quando um atributo ordenado é escolhido, pois o número de valores pode ser muito grande. Geralmente, dado um atributo $a$, tal que $\{v_1, ..., v_m\}$ são seus valores possíveis, procura-se encontrar um limiar $T$ de forma que duas partições $I_1$ e $I_2$ sejam geradas, tal que $[\forall v_i \in I_1, v_i \leq T]$ e $[\forall v_i \in I_2, v_i > T]$. Existem, portanto, $m - 1$ limiares possíveis, e todos precisam ser verificados. Para cada limiar $T_i$ $(1 \leq i \leq m)$ calculado o ganho de informação $G_i$ e o limiar escolhido aquele que maximiza o ganho, ou seja, $T_i$ tal que

$\exists i \forall j (1 \leq i, j \leq m)$ e $(i \neq j)$, $G_i > G_j$.

## 2.4 Poda em Árvores de Decisão

Geralmente, as ADs geradas, a partir do processo descrito nesta Seção, são complexas e superadequam-se aos dados. Esse efeito é derivado do particionamento excessivo do conjunto que acaba por gerar nós-folha não representativos, ou seja, que representam uma parte muito pequena dos dados de treinamento. Árvores com estas características não apresentam boa performance sobre novos dados, pois os conceitos cobrem as peculiaridades do conjunto de treinamento e não representam os conceitos gerais que poderiam ser obtidos indutivamente. O problema pode ser abordado, basicamente, de duas formas:

- *pré-poda*: consiste em evitar a subdivisão do conjunto segundo algum critério, como por exemplo, testanto a taxa de erro, a média do ganho de informação, etc.

- *pós-poda*: consiste em gerar a árvore completa e depois simplificá-la. O processo de simplificação é realizado transformando subárvores em nós-folha. A classe associada a cada novo nó-folha criado é dada pela classe mais freqüente.

Existem vantagens e desvantagens em ambas as técnicas de poda, embora pós-poda seja a mais utilizada. Pré-poda apresenta melhor performance de processamento, uma vez que ramos excessivos não são gerados. Porém, a escolha de um critério para poda não é uma tarefa simples. Pós-poda demonstrou apresentar melhores resultados em termos de classificação [Murphy, 1994], por outro lado, a busca e união de nós envolvidos no processo pode alterar negativamente a performance do sistema.

## 2.5 Classificação Bayesiana com Naïve Bayes

Assim como AD, o paradigma probabilístico [Hanson, 1990] [Mitchell, 1997] tem sido utilizado para a tarefa de classificação e tem obtido resultados positivamente surpreendentes [McCallum, 1998] [Hanson, 1990] [Joachims, 1996]. No aprendizado probabilístico, tenta-se estimar os parâmetros do modelo gerado a partir dos dados. Geralmente, utiliza-se a regra de Bayes (Seção 3.2) para se obter as probabilidades das respectivas hipóteses e a classe com maior significância probabilística é a escolhida.

O classificador Naïve Bayes [McCallum, 1998] [Hanson, 1990] assume independência total entre os atributos que constituem os dados de treinamento. Essa suposição é, obviamente, inválida para aplicações do mundo real, onde geralmente há fortes relacionamentos entre as variáveis. Porém, os resultados obtidos pelo classificador Naïve Bayes desmentem a obrigatoriedade de tais relacionamentos para a tarefa de classificação, pois demonstram altas taxas de precisão.

Dado um conjunto de treinamento $E$, formado por exemplos com formato

$$e =< a_{v_1}, ..., a_{v_m} >$$

tal que, $a_{v_i}$ é o valor para o atributo $a_i$, a probabilidade estimada desse conjunto de valores representar uma hipótese $h$ é dada por:

$$p(e|h) = p(h) \times \prod_{i=1}^{m} P(a_{vi}|h) \tag{2.10}$$

Substituindo-se na regra de Bayes, obtêm-se:

$$p(h|e) = \frac{p(h) \times \prod_{i=1}^{m} P(a_{vi}|h)}{\sum_{j=1}^{k} p(h_j) \times \prod_{i=1}^{m} P(a_{vi}|h_j)} \tag{2.11}$$

Como é possível perceber, a suposição de independência entre os atributos feita por Naïve Bayes define o cálculo da probabilidade de uma evidência conjuntiva $< a_{v_1}, ..., a_{v_m} >$ como sendo o produto das probabilidades individuais de cada atributo [Mitchell, 1997].

O modelo aprendido pelo classificador Naïve Bayes é formado, portanto, pelo conjunto de probabilidades — $p(h)$ e $p(e|h)$ — calculadas a partir dos dados. Diferentemente de outros métodos de aprendizado, pode-se dizer que a busca por uma determinada hipótese não é realizada através do espaço de hipóteses, mas calculando-se a freqüência dos valores no conjunto de treinamento.

Considere-se o Exemplo 2.2.1 — página 9. Dado o exemplo de teste $e$:

| | *Exterior* | *Temperatura* | *Umidade* | *Vento* |
|---|---|---|---|---|
| $e$ | *ensolarado* | *baixa* | *alta* | *sim* |

é necessário prever a sua respectiva classe. De acordo com a abordagem Naïve Bayes, tem-se:

| | $Sim$ | $Não$ |
|---|---|---|
| $p(h)$ | $\frac{9}{14} = 0.643$ | $\frac{5}{14} = 0.357$ |
| $p(exterior = ensolarado)$ | $\frac{2}{9} = 0.222$ | $\frac{3}{5} = 0.6$ |
| $p(temperatura = baixa)$ | $\frac{3}{9} = 0.333$ | $\frac{1}{5} = 0.2$ |
| $p(Umidade = alta)$ | $\frac{3}{9} = 0.333$ | $\frac{4}{5} = 0.8$ |
| $p(vento = sim)$ | $\frac{3}{9} = 0.333$ | $\frac{3}{5} = 0.6$ |
| $p(e\|h)$ | $\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} = 0.0053$ | $\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} = 0.0206$ |

Baseado no modelo de aprendizado Naïve Bayes, pode-se concluir que a classe associada ao exemplo $e$ é *Não*, pois sua probabilidade — $p(e|h)$ — é maior.

Substituindo $p(e|h)$ na regra de Bayes é possível calcular ambas as probabilidades condicionais:

$$p(Sim|e) = \frac{\frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}}{\left(\frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}\right) + \left(\frac{5}{4} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}\right)} = 0.2046$$

$$p(Não|e) = \frac{\frac{5}{4} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}}{\left(\frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}\right) + \left(\frac{5}{4} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}\right)} = 0.7954$$

## 2.6 Considerações Finais

No Apêndice A.2 são apresentados os principais conceitos de AM. Dentre os diversos paradigmas de AM, dois deles foram abordados neste Capítulo. O primeiro, Aprendizado Indutivo Baseado em Árvores de Decisão, é um dos mais difundidos.

Verificou-se que algoritmos de AD são capazes de generalizar conceitos eficientemente a partir dos dados. Dentre as principais vantagens de AD estão a rapidez e compreensibilidade do conhecimento adquirido e a simplicidade dos algoritmos. Foram discutidas, ainda, diversas peculiaridades de tais algoritmos e algumas questões técnicas mais importantes.

Foi apresentada, de maneira sucinta, uma outra abordagem de AM, baseada no paradigma probabilístico — o classificador Naïve Bayes. O classificador Naïve Bayes pressupõe que a probabilidade de uma evidência conjuntiva $e =< a_{v1}, ..., a_{vm} >$ pertencer a uma hipótese $h$ é dada pelo produto da probabilidade da ocorrência de cada um dos valores de seus atributos, uma vez que os atributos são considerados independentemente. Apesar da suposição de independência não ser verdadeira para a maioria dos domínios de aplicação, verifica-se em diversos trabalhos que a classificação produzida pelo Naïve Bayes possui alta precisão.

Algoritmos de AM, geralmente, são sensíveis a imperfeições nos dados, como por exemplo, exemplos com valores faltando para atributos, erros de amostragem,

falta de atributos relevantes, inconsistência entre os exemplos, etc. A presença de informações imperfeitas, ou incompletas, exige que técnicas não tradicionais sejam empregadas na inferência sobre tais informações. Em IA, técnicas de Gerenciamento de Incerteza podem ser utilizadas com o objetivo de se obter inferências confiáveis sobre informações imperfeitas. Algumas dessas técnicas serão abordadas neste trabalho posteriormente.

# Capítulo 3

# Gerenciamento de Incerteza

## 3.1 Considerações Iniciais

A solução de grande parte dos problemas de IA necessita que conhecimento, bem como quaisquer outras informações pertinentes, estejam disponíveis. Porém, nem sempre a informação necessária para a solução de determinado problema está disponível de forma completa e consistente. Portanto, torna-se necessária a utilização de técnicas de Gerenciamento de Incerteza — GI — que possibilitem o raciocínio sobre informações com tais características [McClean, 1998] [Wolkenhauer, 1997]. Situações do mundo real estão repletas de incertezas, e, geralmente, são resolvidas de forma intuitiva, o que não significa, necessariamente, obter uma solução correta.

Um exemplo de situação onde incerteza está implicitamente envolvida: um engenheiro que trabalha em uma companhia de extração petrolífera recebe a descrição geográfica detalhada de determinada área e, com o objetivo de encontrar petróleo, decide perfurá-la. *Como garantir que a área de extração escolhida é a área onde há a maior concentração de petróleo?* Essa questão é necessária, uma vez que a análise realizada poderia não estar suficientemente detalhada. Além disso, de posse de tal análise, o engenheiro poderia fornecer um parecer errado, seja por inexperiência ou

falta de conhecimento.

> "Conhecimento incerto é o conhecimento que não é indiscutível, mas ao qual está associada alguma medida de incerteza que descreve crenças para as quais existem certas evidências de apoio [Rich, 1994]".

Em [Ladeira, 1996] são apresentadas as seguintes formas possíveis de deficiência no conhecimento:

- *o conhecimento é parcial*: intuitivamente, o conhecimento é dito parcial quando informações importantes para a solução de um problema relacionado não estão disponíveis;
- *o conhecimento é não completamente confiável*: diversos podem ser os motivos que conduzem o conhecimento a não tornar-se completamente confiável. Por exemplo, durante a geração de uma base de dados erros de digitação podem ser cometidos, etc;
- *a linguagem de representação é imprecisa*: a linguagem utilizada para a representação do conhecimento necessita ser clara, uma vez que interpretações equivocadas podem levar a inferências erradas;
- *o conhecimento é conflitante*: informações conflitantes são comuns tanto em bases de dados, quanto em bases de conhecimento, pois quando mais de uma fonte está envolvida no processo de criação e aquisição de informações podem surgir opiniões controversas.

Nesta Seção serão apresentadas as principais técnicas empregadas no raciocínio sobre informações incertas, dentre elas, Teoria da Probabilidade (Teoria Bayesiana), Teoria da Evidência de Dempster-Shafer, Fatores de Certeza e Lógica Fuzzy.

## 3.2 Teoria da Probabilidade

A Teoria Moderna da Probabilidade teve seu desenvolvimento iniciado por volta do ano 1600 [Henkind, 1988] [Ng, 1990]. Atualmente, probabilidades são utilizadas por pesquisadores de Inteligência Artificial na solução de diversos problemas, como manipulação de informações incertas em Sistemas Especialistas [Buchanan, 1984] [Pearl, 1988] [Marcus, 1986] e classificação em sistemas de Aprendizado de Máquina e Data Mining [Ávila, 1998] [Clark, 1989] [Quinlan, 1993].

Em Teoria da Probabilidade — TP — é assumida total aleatoriedade entre as ocorrências de determinado evento $A$. A probabilidade de $A$, denotada $p(A)$, consiste, portanto, na proporção de casos em que $A$ ocorre. O conjunto de todos os eventos do mundo, espaço de amostra, ou, simplesmente, espaço de eventos, é denotado por $\Omega$. A função $p$, portanto, deve satisfazer três axiomas:

1. a probabilidade de qualquer evento $A$ é não-negativa, ou seja: $\forall A \in \Omega\ p(A) \geq 0$;
2. a probabilidade do espaço de amostra $\Omega$ é 1, ou $p(\Omega) = 1$;
3. se $n$ eventos são mutuamente exclusivos, então a probabilidade de ocorrência de pelo menos um desses eventos é a soma das probabilidades individuais desses eventos, ou $p(A_1 \cup ... \cup A_n) = \sum_{i=1}^{n} p(A_i)$.

Com base nos axiomas anteriores, pode-se deduzir que

$$\forall A \in \Omega\ 0 \leq p(A) \leq 1 \tag{3.1}$$

Isso significa que para qualquer evento $A$, a $p(A)$ está entre 0 e 1. Logo, quando $p(A) = 0$ significa que $A$ nunca ocorreu, caso contrário, $p(A) = 1$ significa que apenas o evento $A$ ocorre. O complemento de $A$ ($\neg A$) representa todos os eventos

pertencentes a $\Omega$, exceto $A$. Uma vez que $A$ e $\neg A$ são mutuamente exclusivos, e $A \cup \neg A = \Omega$, baseado no axioma 3 tem-se:

$$p(A) + p(\neg A) = p(A \cup \neg A) = p(\Omega) = 1 \tag{3.2}$$

A Equação 3.2 é equivalente a $p(\neg A) = 1 - p(A)$, dessa forma, obtem-se, facilmente, o valor da probabilidade de $\neg A$.

Assumindo que $B$ é um outro evento, a probabilidade de $A$ ocorrer, dado que $B$ ocorre, denotada $p(A|B)$ é chamada a probabilidade condicional de $A$ dado $B$. A probabilidade que ambos $A$ e $B$ ocorram, $p(A \cap B)$, é chamada a probabilidade conjunta de $A$ e $B$. Por definição, a $p(A|B)$ é calculada através da divisão da probabilidade conjunta de A e B, $p(A \cap B)$, pela probabilidade de $B$, $p(B)$:

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{3.3}$$

É importante notar que quando é assumida a ocorrência de $B$, $p(B)$ é diferente de zero.

Seguindo a definição temos que a probabilidade condicional de $B$ dado $A$ é:

$$p(B|A) = \frac{p(A \cap B)}{p(A)} \tag{3.4}$$

Uma vez que a probabilidade condicional é comutativa, ou seja, $p(B \cap A) = p(A \cap B)$, tem-se:

$$p(A \cap B) = p(B \cap A) = p(B|A) \times p(A) \tag{3.5}$$

Substituindo a expressão 3.5 em 3.3, é obtida a regra de Bayes:

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B)} \tag{3.6}$$

Quando é assumida a independência entre dois eventos (a ocorrência de um evento não afeta a ocorrência do outro), então, por definição, $p(A|B) = p(A)$ e $p(B|A) = p(B)$. Relacionando probabilidades e Teoria dos Conjuntos [Abe, 1992], se $A$ e $B$ são conjuntos disjuntos, a união desses conjuntos corresponde à soma das probabilidades de $A$ e $B$ e a interseção corresponde à multiplicação entre as probabilidades. Dessa forma, se $A$ e $B$ são conjuntos disjuntos, $p(A \cup B) = p(A) + p(B)$ e $p(A \cap B) = p(A) \times p(B)$.

Prosseguindo com a teoria dos conjuntos, o conjunto $B$ pode ser descrito como $(B \cap A) \cup (B \cap \neg A)$. Como a união é disjunta:

$$\begin{aligned} p(B) &= p((B \cap A) \cup (B \cap \neg A)) \\ &= p(B \cap A) + p(B \cap \neg A) \\ &= p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A) \end{aligned} \tag{3.7}$$

Combinando as equações 3.6 e 3.7:

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A)} \tag{3.8}$$

A Equação 3.8 permite o cálculo da probabilidade condicional de $A$ dado $B$ a partir de $p(B|A)$. Essa pode ser uma característica importante da Teoria da Probabilidade em GI. Considerando uma base de conhecimento formada por regras com o seguinte formato:

*se* $H$ é verdade, *então* $E$ é verdade com probabilidade $p$

a Equação 3.8 fornece uma forma de computar a probabilidade de $H$ — $p$ —, caso

ela não seja fornecida.

Substituindo as notações de $A$ por $H$ — simbolizando uma hipótese — e $B$ por $E$ — um item de evidência, a Equação 3.8 é apresentada da seguinte forma:

$$p(H|E) = \frac{p(E|H) \times p(H)}{p(E|H) \times p(H) + p(E|\neg H) \times p(\neg H)} \tag{3.9}$$

tal que $p(H)$ é a probabilidade *a priori* da hipótese $H$. A equação 3.9 relaciona uma hipótese a um item de evidência e também relaciona uma evidência observada a uma hipótese produzida não substancialmente. Ela permite que, dada uma hipótese individual $H$ e uma evidência $E$, seja calculada a probabilidade posterior associada a $H$ sobre a observação da evidência $E$. Porém, ela pode ser generalizada de forma a permitir o cálculo da probabilidade posterior para múltiplas hipóteses da seguinte forma:

$$p(H_i|E) = \frac{p(E|H_i) \times p(H_i)}{\sum_{k=1}^{m} p(E|H_k) \times p(H_k)} \tag{3.10}$$

Como foi visto, assumindo que $E_1$ e $E_2$ são independentes, $p(E_1E_2|H) = p(E_1|H) \times p(E_2|H)$, pode-se definir

$$p(H_i|E_{1...}E_n) = \frac{p(E_{1...}E_n|H_i) \times p(H_i)}{\sum_{k=1}^{m}(p(E_{1...}E_n|H_k) \times p(H_k))} \tag{3.11}$$

O Apêndice B.1 apresenta um pequeno exemplo de uma aplicação da TP no raciocínio incerto.

## 3.3 Teoria da Evidência de Dempster-Shafer

A Teoria da Evidência de Dempster-Shafer — DST — [Henkind, 1988] [Ng, 1990] foi desenvolvida por Dempster nos anos 60 e extendida por Glen Shafer nos anos 70. As

motivações para o desenvolvimento de tal teoria são algumas limitações encontradas em TP:

- como representar ignorância sobre determinada informação: é importante relembrar o significado de $p(A) = 0$ para que não sejam geradas conclusões equivocadas. Quando se afirma que a probabilidade de um evento é zero, se está afirmando a absoluta falta do evento $A$, ou seja, sabe-se algo sobre $A$. A forma mais comum de expressar ignorância em TP é associar uma probabilidade uniforme a todas as hipóteses. Porém, esta estratégia sugere indiretamente ignorância sobre todas as hipóteses, o que pode não ser verdadeiro;

- a crença associada a um determinado evento somada à negação dessa crença, não, necessariamente, é igual a um.

A DST tem recebido um crescente interesse da comunidade de IA, pois tem se mostrado uma ferramenta potencialmente poderosa no tratamento de domínios incertos de aplicação [Henkind, 1988].

O conjunto total de preposições mutuamente exclusivas sobre um determinado domínio, denotado $\theta$, é chamado *estrutura de discernimento* [Ma, 1995]. O conjunto de todos os subconjuntos de $\theta$, $2^\theta$, é formado pelas proposições do domínio. Para cada subconjunto de $2^\theta$ é atribuído um intervalo $[Cr, Pl]$, tal que $Cr$ representa a força da evidência em favor ao subconjunto e varia de 0 (falta de evidências) até 1 (certeza absoluta). O símbolo $Pl$ (plausibilidade) é definido como:

$$Pl(s) = 1 - Cr(\neg s)$$

tal que, $s$ é um subconjunto de proposições [Ng, 1990] [Ladeira, 1996] [Russel, 1995] [Klopotek, 1997]. A plausibilidade também varia de 0 a 1 e mede até que ponto a evidência em favor de $\neg s$ deixa espaço em favor da crença em $s$.

Se há evidências totais em relação a $\neg s$, então $Cr(\neg s)$ será 1, logo, $Pl(s) = 0$. Portanto, o único valor possível para $Cr(s)$ também é 0 [Rich, 1994].

De maneira semelhante à TP, há um espaço de hipóteses — $\Omega$ em TP — denotado $\theta$ em DST. Porém em DST, o tamanho do espaço é dado por $2^{|\theta|}$, enquanto que em TP o número de hipóteses é dado por $|\Omega|$ [Ng, 1990].

Duas funções básicas adicionais necessárias à DST necessitam ser definidas:

- $m$: função básica de probabilidade;
- $Cr$: função de crença.

A função $m(A)$ representa a medida de probabilidade do conjunto $A$ de hipóteses ser o conjunto verdadeiro, e possui as seguintes propriedades:

$$m(\phi) = 0$$

e

$$\sum_{A \subseteq \theta} m(A) = 1 \qquad para\ A \subseteq \theta$$

A função $Cr(A)$, dada pela fórmula

$$Cr(A) = \sum_{B \subseteq A} m(B) \qquad para\ A \subseteq \theta \tag{3.12}$$

representa a crença global em que a hipótese desejada esteja presente no conjunto $A$.

A soma das crenças de $A$ e $\neg A$ deve ser menor ou igual a 1, ou seja, $Cr(A) + Cr(\neg A) \leq 1$. Isso ocorre devido a definição de uma hipótese e sua negação. Em DST, $A$ significa "$A$ e apenas $A$", enquanto que $\neg A$ significa "tudo menos $A$", ou $\theta - A$ [Ng, 1990].

Segundo a definição, dados dois conjuntos de hipóteses, $A = \{a, b\}$ e $B = \{c\}$, as funções $Cr(A)$ e $Cr(B)$ são obtidas da seguinte forma:

$$\begin{aligned} Cr(A) &= m(a) + m(b) + m(\{a, b\}) \\ Cr(B) &= m(c) \end{aligned}$$

O Apêndice B.2 apresenta um exemplo da aplicação da Teoria de Dempster-Shafer.

## 3.4 Fatores de Certeza

Durante o desenvolvimento da tecnologia de sistemas especialistas, deparou-se com a necessidade de manipulação e raciocínio sobre informações incertas. Aplicações médicas são exemplos de domínios onde incerteza é praticamente constante em todos os procedimentos de raciocínio. Esta necessidade impulsionou o desenvolvimento de técnicas probabilísticas capazes de manipular bases incertas de conhecimento. O primeiro sistema desenvolvido com tal tecnologia foi o MYCIN [Buchanan, 1984] [Neapolitan, 1990], um sistema especialista que recomenda terapias para pacientes com infecções bacteriológicas. A base de conhecimento de tal sistema é representada na forma de regras que possuem o seguinte formato:

$$se \ < evidência > \ então \ FC \ < hipótese >$$

tal que, $FC$ denota a crença em $< hipótese >$, dado que a evidência $< evidência >$ é observada [Ng, 1990].

Para que as propagações de evidências possam ocorrer, duas funções necessitam ser calculadas, são elas:

- $MC[h, e]$: mede o grau do aumento da crença na hipótese $h$, uma vez que a evidência $e$ é observada;

| $[-1,+1]$ | $[0,+1]$ |
|---|---|
| -1 | 0 |
| -0.75 | 0.125 |
| -0.5 | 0.25 |
| -0.25 | 0.375 |
| 0 | 0.5 |
| 0.25 | 0.625 |
| 0.5 | 0.75 |
| 0.75 | 0.875 |
| 1 | 1 |

Tabela 3.1: Normalização de $[-1,+1]$ para $[0,+1]$.

- $MD[h,e]$: mede o grau do aumento da descrença na hipótese $h$, uma vez que a evidência $e$ é observada.

Ambas as funções anteriores representam valores numéricos pertencentes ao intervalo [0,1]. A partir destas duas medidas, pode-se definir uma única medida denominada fator de certeza — FC — da seguinte forma:

$$FC[h,e] = MC[h,e] - MD[h,e] \tag{3.13}$$

Seguindo a definição, cada fator de certeza associado a uma regra pode assumir valores pertencentes ao intervalo $[-1,+1]$. Valores positivos, indicam crença favorável à hipótese, por outro lado, valores negativos indicam crença favorável à negação da hipótese. Intervalos diferentes de $[-1,+1]$ podem ser definidos, pois trata-se apenas de uma questão notacional. Por exemplo, caso o intervalo desejado seja $[0,1]$, basta aplicar a equação de normalização $FC/2 + 0.5$, como mostrado na Tabela 3.1.

As evidências associadas às regras de uma base de conhecimento com o formato apresentado anteriormente, geralmente, necessitam ser combinadas para que a solução

de um determinado problema seja encontrada. Em [Rich, 1994], são definidas três formas de combinação entre duas regras. Elas são ilustradas na Figura 3.1.

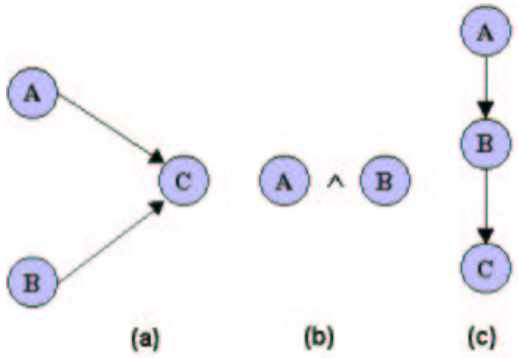



Figura 3.1: Formas de Combinação de Regras.

Na Figura 3.1 (a), várias evidências são combinadas para determinar o FC de apenas uma determinada hipótese. Dadas duas observações $e_1$ e $e_2$, as medidas de crença e descrença de uma hipótese são calculadas da seguinte forma:

$$MC[h, e1 \wedge e2] = \begin{cases} 0 & \text{se MD[h,e1}\wedge\text{ e2]=1} \\ MC[h, e_1] + MC[h, e_2] \times (1 - MC[h, e_1]) & \textit{caso contrário} \end{cases} \quad (3.14)$$

$$MD[h, e1 \wedge e2] = \begin{cases} 0 & \text{se MC[h,e1}\wedge\text{ e2]=1} \\ MD[h, e_1] + MD[h, e_2] \times (1 - MD[h, e_1]) & \textit{caso contrário} \end{cases} \quad (3.15)$$

A Figura 3.1 (b) representa a situação onde várias hipóteses são combinadas, dada apenas uma observação — evidência. Este caso ocorre, geralmente, quando se necessita calcular o $FC$ do antecedente de uma regra que contenha várias cláusulas, tal que cada cláusula corresponde à uma hipótese. No MYCIN, as regras de combi-

nação da conjunção e disjunção de hipóteses são as seguintes [Rich, 1994]:

$$MC[h_1 \wedge h_2, e] = min(MC[h_1, e], MC[h_2, e]) \quad (3.16)$$

$$MC[h_1 \vee h_2, e] = max(MC[h_1, e], MC[h_2, e]) \quad (3.17)$$

$$e$$

$$MD[h_1 \wedge h_2, e] = min(MD[h_1, e], MD[h_2, e]) \quad (3.18)$$

$$MD[h_1 \vee h_2, e] = max(MD[h_1, e], MD[h_2, e]) \quad (3.19)$$

A situação representada na Figura 3.1 (c) sugere que o encadeamento das regras é necessário, ou seja, o resultado da incerteza de uma regra serve como entrada para outra regra. Isto pode ocorrer, por exemplo, durante o cálculo inicial de incertezas de regras que representam testes de laboratório, onde a execução de cada teste influencia o resultado do próximo teste. A regra de encadeamento definida no MYCIN é apresentada da seguinte forma:

$$MC[h, e_1] = MC\prime[h, e_1] \times \max(0, FC[e_1, e]) \quad (3.20)$$

tal que, $MC\prime[h, e_1]$ é a medida de crença atribuída a $h$, uma vez que $e_1$ é válida.

Dada a seguinte regra:

| | | | |
|---|---|---|---|
| (1) | **se** | pigmento é gram-positivo | e |
| (2) | | morfologia é coccus | e |
| (3) | | crescimento é em blocos | |
| (h) | **então** | a identidade do organismo é estafilococo (0.7) | |

pode-se assumir que o valor de certeza da regra é 0.7 em relação às evidências (1), (2) e (3). No MYCIN, assim como na maioria dos sistemas especialistas que utilizam fatores de certeza, os valores iniciais de certeza são fornecidos pelo especialista durante a geração da base de conhecimento. Pode-se dizer, portanto, que 0.7 é a probabili-

dade $p(h)$ com que o especialista acredita na hipótese $h$. Logo, $1 - p(h)$, pode ser considerada como a estimativa do especialista na descrença em $h$ [Buchanan, 1984].

Se $p(h|e)$ é maior que $p(h)$, então a evidência $e$ aumenta a crença do especialista em $h$ e diminui a descrença, considerando a verdade de $h$. Portanto, o aumento proporcional na crença é calculada da seguinte forma:

$$MC[h, e] = \frac{p(h|e) - p(h)}{1 - p(h)} \tag{3.21}$$

A equação anterior calcula a medida de crença aumentada em $h$, resultante da observação de $e$.

Outra situação possível, é aquela tal que $p(h|e)$ é menor que $p(h)$. A observação de $e$, portanto, deve diminuir a crença na hipótese $h$ e aumentar a descrença em $h$. A diminuição proporcional na crença em $h$ é dada da seguinte forma:

$$MD[h, e] = \frac{p(h) - p(h|e)}{p(h)} \tag{3.22}$$

Esta equação mede a descrença aumentada em $h$, dada a observação $e$.

Pode-se, conceitualmente, traduzir as funções anteriormente definidas da seguinte maneira:

- $MC[h, e]$: é o aumento proporional da crença em $h$, a partir da evidência observada $e$;
- $MD[h, e]$: é a diminuição proporcional da crença na hipótese $h$, resultante da observação de $e$;
- $p(h)$: é a crença na hipótese $h$ em qualquer momento;
- $1 - p(h)$: é a descrença estimada sobre $h$ em qualquer momento.

É importante lembrar que um determinado item de evidência $e$ não pode favorecer e desfavorecer uma hipótese ao mesmo tempo. Portanto, quando $MC[h,e] > 0$, então $MD[h,e] = 0$, e quando $MD[h,e] > 0$, $MC[h,e] = 0$.

A partir desse ponto, as definições anteriores podem ser definidas, formalmente, em termos de probabilidades *condicionais* e *a priori* (Seção 3.2):

$$MC[h,e] = \begin{cases} 1 & \text{se} \quad \text{p(h)=1} \\ Large\frac{max[p(h|e),p(h)]-p(h)}{1-p(h)} & caso\ contrário \end{cases} \tag{3.23}$$

$$\begin{cases} 1 & \text{if} \quad \text{p(h)=0} \\ \frac{min[p(h|e),p(h)]-p(h)}{-p(h)} & caso\ contrário \end{cases} \tag{3.24}$$

Assim como na Teoria da Probabilidade — TP — e Teoria de Dempster-Shafer — DST —, é necessário definir uma forma de combinação entre duas regras que correspondem à mesma evidência. Neste caso, os fatores de certeza $FCs$ das regras são combinados da seguinte forma:

$$\begin{aligned} FC_{comb}(FC_1, FC_2) &= FC_1 + FC_2 - FC_1 \times FC_2 && \text{se } FC_1 \text{ e } FC_2 > 0 \\ &= (FC_1 + FC_2)/(1 - \min(|FC_1|, |FC_2|)) && \text{se } FC_1 \text{ ou } FC_2 < 0 \\ &= FC_1 + FC_2 + FC_1 \times FC_2 && \text{se } FC_1 \text{ e } FC_2 < 0 \end{aligned}$$

O valor combinado obtido anteriormente pode ser representado como:

$$FC_{comb}[h, e_1 \wedge e_2] \tag{3.25}$$

tal que, $(e_1 \wedge e_2)$ representa a conjunção de tais evidências. A forma de combinação de regras que representam várias hipóteses e possuem a mesma evidência é dada a seguir:

$$FC\prime_{comb}[h_1 \wedge h_2, e] = \min(FC[h_1, e], FC[h_2, e]) \tag{3.26}$$

As duas últimas funções de combinação apresentadas ($FC_{comb}$ e $FC\prime_{comb}$), foram utilizadas pelo sistema especialista EMYCIN [Ng, 1990] [Henkind, 1988] com o objetivo de criar uma forma de combinação que utilizasse, diretamente, apenas um valor numérico ($FC$) e não dois valores ($MC$ e $MD$). Além disso, as funções de combinação do EMYCIN superam algumas dificuldades — variações de crenças incorretas através da supervalorização de evidências [Russel, 1995] [Ladeira, 1996] — encontradas na proposta inicial do MYCIN.

## 3.5 Lógica Fuzzy

Os primeiros trabalhos sobre Lógica Fuzzy foram desenvolvidos por Zadeh em 1965 [Zadeh, 1965]. Posteriormente, inúmeras propriedades dos conjuntos Fuzzy foram estudadas e problemas de diferentes domínios puderam ser modelados através da Teoria Fuzzy. Pode-se utilizar a teoria dos conjuntos fuzzy para manipulação de incerteza em problemas de IA [Henkind, 1988].

Dentre os objetivos da Lógica Fuzzy está o tratamento mais adequado para determinadas situações onde a teoria clássica dos conjuntos [Abe, 1992] apresenta evidentes deficiências. Por exemplo, dado um determinado problema que utiliza o conceito de "homem alto". Poder-se-ia definir que o conjunto "homem alto" é formado por todos os homens que possuem altura maior ou igual a 1.80 m. Mas, visualmente, não poderia ser dito que um homem que possui altura de 1.795 m é alto? Segundo a teoria clássica dos conjuntos — conjuntos Crisp — isto não é possível.

Antes de apresentar os principais conceitos de Lógica Fuzzy, serão recordados alguns conceitos da teoria clássica dos conjuntos.

Dado que $X$ denota um conjunto de elementos, $x$ denota um elemento individual de $X$. $A$ representa um subconjunto de $X$ e $\mu_A$ é uma função característica — ou

| $x$ | altura (m) | $\mu_T(x)$ | Significado |
|---|---|---|---|
| João | 1.80 | 1.0 | João é alto |
| Pedro | 1.70 | 0.5 | Pedro é meio alto |
| Paulo | 1.55 | 0.0 | Paulo não é alto |

Tabela 3.2: Função de Pertinência.

função de pertinência — de $A$ se e somente se:

$$\mu_A(X) = \begin{cases} 1 & \textit{se } X \in A \\ 0 & \textit{caso contrário} \end{cases}$$

Geralmente, o conjunto {0,1} é chamado o conjunto de valoração e o valor associado pela função $\mu$, para um dado $x$, é chamado grau de pertinência. Portanto, os valores possíveis para uma função de pertinência — em conjuntos crisp — são 1 e 0. Por exemplo, assumindo $X$ ser o conjunto de números inteiros e $C$ o conjunto de números múltiplos de 5, então:

$$\mu_C(x) = 1 \quad \text{para} \quad x = 35$$

e

$$\mu_C(x) = 0 \quad \text{para} \quad x = 27$$

Na teoria dos conjuntos Fuzzy, o conjunto de valoração é extendido de {0,1} para o intervalo [0,1], ou seja, o grau de pertinência de um determinado elemento $x$ pode assumir qualquer valor numérico pertencente a [0,1]. Intuitivamente, $\mu_C(x) = 1$ é equivalente a $x \in C$. Por exemplo, seja $X$ o conjunto de todas as pessoas e $T$ o conjunto de todas as pessoas altas, a Tabela 3.2 descreve a função $\mu_T(x)$.

A utilização de técnicas dos conjuntos Fuzzy depende da definição de uma função característica para cada conjunto empregado. Por exemplo, a função $\mu_T(x)$, descrita anteriormente, poderia ser definida como apresentada na Tabela 3.3.

| $\mu_T(x)$ | |
|---|---|
| 0 | para $altura(x) \leq 1.60$ |
| $(altura(x) - 1.60)/2$ | para $1.60 \leq altura(x) \leq 1.80$ |
| 1 | para $altura(x) \geq 1.80$ |

Tabela 3.3: Definição da Função de Pertinência.

Uma característica importante, inerente às funções de pertinência, é a sensibilidade ao contexto do problema. Por exemplo, se o universo de pessoas do conjunto $X$ representasse jóckeis de turf, então, talvez, o valor de $\mu_T(Pedro)$ seria mais adequado se fosse igual a 1.

Diferentemente das técnicas de GI apresentadas nas Seções anteriores, Lógica Fuzzy não está relacionada à probabilidades. Portanto, os valores obtidos através de uma função $\mu_D(x)$ não representam a probabilidade de $x$ pertencer ao conjunto $D$, e sim o grau de pertinência de $x$ em relação a $D$.

As duas operações principais definidas por Zadeh são a interseção $\cap$ e a união $\cup$. Ambas são definidas da seguinte forma:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{3.27}$$

$$e$$

$$\mu_{A \cup B}(x) = max(\mu_A(x), \mu_B(x)) \tag{3.28}$$

Muitas outras definições foram propostas para as operações de interseção e união, geralmente, com o objetivo de satisfazer certas proposições algébricas como monotonicidade, comutatividade, e associatividade [Henkind, 1988]. Para tanto, porém, foram definidas normas triangulares ou normas $T$ para a formulação da operação de interseção e *co-normas* $T$ para formulação da união, pois nem todos os trabalhos utilizam a formulação de *min* e *max* para interseção e união, respectivamente.

Pode-se relacionar a teoria de Lógica Fuzzy com lógicas multi-valoradas —

lógicas tais que o valor-verdade de uma proposição pode assumir infinitos valores [Ávila, 1996] [Subrahmanian, 1987a] [Subrahmanian, 1987b]. Além disso, pode-se utilizar qualificadores lingüísticos para aumentar o conjunto de valores de uma função de pertinência. Porém, a utilização de qualificadores implica na definição explícita de cada um deles para uma determinada variável.

A Figura 3.2 representa graficamente a definição de uma variável chamada temperatura que possui três qualificadores lingüísticos: baixa, média e alta.

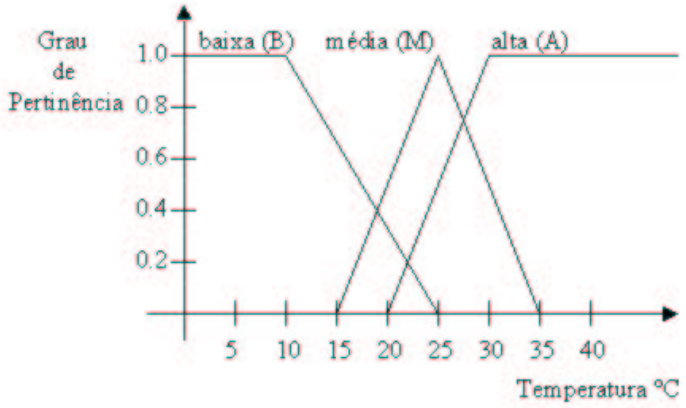



Figura 3.2: Definição da Variável Temperatura.

De acordo com a definição de conjuntos Fuzzy, as operações de união e interseção entre os conjuntos $B$ e $M$, ilustrados na Figura 3.2, são representadas nas Figuras 3.3 e 3.4, respectivamente.

## 3.6 Considerações Finais

Nesta Seção procurou-se abordar o problema de informações incertas, inerente a diversos domínios de aplicação. Estudos realizados em [Rich, 1994] [Sandri, 1996] [Ladeira, 1996] indicam que várias podem ser as causas de incerteza em sistemas computacionais de Inteligência Artificial. Dentre elas estão a presença de infor-

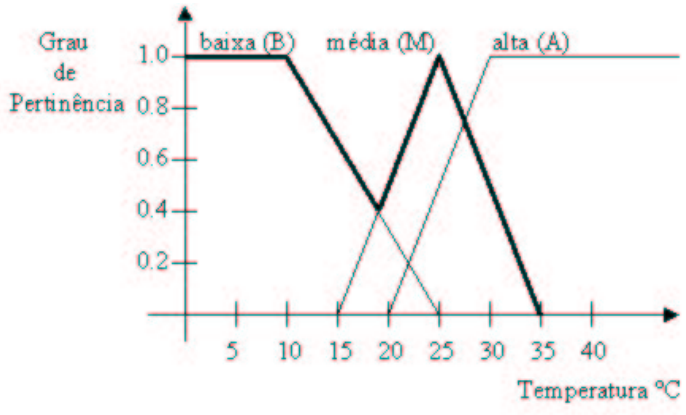



Figura 3.3: União Entre os Conjuntos B e M.

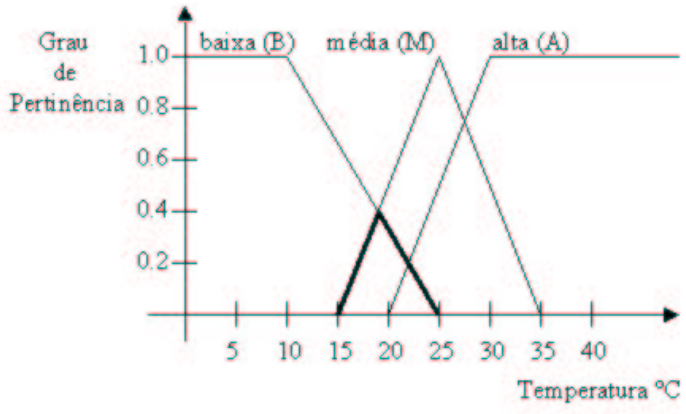



Figura 3.4: Interseção Entre os Conjuntos B e M.

mações imprecisas, informações probabilísticas e inconsistentes. Este último tipo de imperfeição será abordado de maneira especial, neste trabalho, durante as Seções posteriores.

A partir da caracterização de incerteza, alguns métodos probabilísticos foram introduzidos — Teoria da Probabilidade, Teoria Evidencial de Dempster-Shafer e Fatores de Certeza —, além de conceitos básicos de Lógica Fuzzy.

Todas as abordagens apresentadas possuem pontos fortes e deficiências. Pode-se concluir que todas as técnicas de GI abordadas possuem vantagens e desvantagens, portanto, a aplicação de determinado método sobre um domínio específico, depende

de um profundo estudo sobre o problema, levando-se em conta a complexidade, possíveis causas da incerteza e objetivos a serem alcançados. Além disso, técnicas híbridas também podem ser utilizadas [Yu, 1997].

Nenhuma das formas de GI apresentadas neste Capítulo, foi desenvolvida para manipular especificamente a presença de informações inconsistentes. A inconsistência pode ocorrer, por exemplo, quando informações geradas a partir de fontes distribuídas são unidas para que inferências possam ser realizadas. Com o objetivo de fornecer interpretações adequadas a informações conflitantes, foi desenvolvida a Lógica Paraconsistente [da Costa, 1991] [Blair, 1987] [Subrahmanian, 1987a]. Tal lógica também permite que outros tipos de imperfeições, como por exemplo, ignorância, sejam tratadas de maneira mais adequada, como poderá ser verificado nas Seções posteriores deste trabalho.

# Capítulo 4

# Programação Lógica Paraconsistente

## 4.1 Considerações Iniciais

A presença de informações inconsistentes em sistemas computadorizados pode facilmente ser observada, uma vez que, em diversas aplicações, a inconsistência é inerente ao problema. Para tais sistemas, metodologias de raciocínio quantitativo foram desenvolvidas [Sandri, 1997] [Blair, 1987] [van Emden, 1986] [Subrahmanian, 1987a] [Subrahmanian, 1987b]. Verificou-se, porém, que um sistema lógico estendido, ver [Blair, 1988] [Ng, 1992] [da Costa, 1991] [da Costa, 1963] [da Costa, 1974], poderia fornecer interpretações para informações que são contraditórias — $p$ e $\neg p$. A presença de tais interpretações poderia apresentar, de forma explícita, a inconsistência nos dados e expressar informações importantes para a tomada de decisão.

Um exemplo clássico da literatura, onde é apresentada uma situação de inconsistência, é o diagnóstico médico. Um médico informa ao paciente que ele possui um tumor maligno. Em pânico, este paciente procura a opinião de outro médico, e recebe a informação de que o tumor é benígno. A inconsistência é clara entre

as opiniões dos médicos, e, decisiva para que o paciente tome alguma atitude, por exemplo, procurar um outro médico. Portanto, a eliminação de premissas inconsistentes, implementada por muitos sistemas, nem sempre é a melhor alternativa, pois informações importantes podem ser eliminadas.

A Lógica Paraconsistente procura não eliminar inicialmente as premissas inconsistentes e fornece mecanismos para o raciocínio sobre informações com essa característica. Em [Ávila, 1996], qualifica-se a teoria dos conjuntos anotados — que utilizam valores-verdade explicitamente representados — utilizados pela Lógica Paraconsistente, como extremamente forte, incluindo como caso particular a Lógica Fuzzy.

## 4.2 Programação Lógica Evidencial Paraconsistente

Nesta Seção são apresentados os conceitos de raciocínio lógico evidencial através da Programação Lógica Evidencial — PrLE. Diferentemente de outros trabalhos desenvolvidos sobre raciocínio quantitativo, os valores-verdade utilizados nesta teoria são compostos por dois fatores evidenciais pertencentes a um reticulado $\{x \in \Re | 0 \leq x \leq 1\} \times \{x \in \Re | 0 \leq x \leq 1\}$. A cada item de conhecimento de um sistema lógico evidencial são associados fatores evidenciais de *Crença* e *Descrença*. Intuitivamente, *Crença* representa a quantidade de crença que apóia a verdade da evidência. Por outro lado, *Descrença* representa a quantidade de descrença associada à verdade da evidência, ou a crença associada à falsidade da evidência. Ambos os fatores pertencem ao intervalo [0,1]. Ou seja, infinitos valores podem estar associados às premissas do sistema. Portanto, pode ser definido formalmente um reticulado infinito $\mathcal{T} =< |\mathcal{T}|, \leq >$ tal que

$$|\mathcal{T}| = \{x \in \Re | 0 \leq x \leq 1\} \times \{x \in \Re | 0 \leq x \leq 1\}$$

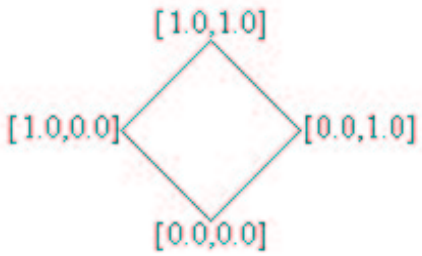



Figura 4.1: Reticulado Infinitamente Valorado.

O reticulado $\mathcal{T}$ ilustrado pelo diagrama de Hasse da Figura 4.1 possui um ponto máximo [1,1]. Este ponto representa a situação de inconsistência máxima, pois acredita-se tanto na verdade quanto na falsidade da premissa em um mesmo instante de tempo. Intuitivamente, a verdade de uma premissa é representada pela evidência [1.0,0.0], pois acredita-se totalmente na verdade e nada é conhecido sobre a falsidade da premissa. Por outro lado, a falsidade é representada por [0.0,1.0], pois acredita-se totalmente na falsidade e nada é conhecido sobre a verdade da premissa. Além da interpretação de inconsistência, um outro conceito introduzido pela Programação Lógica Evidencial Paraconsistente, em relação à Lógica Clássica, é a interpretação de desconhecido — [0.0,0.0]. Neste caso, não se tem informação nem sobre a verdade nem sobre a falsidade do valor-verdade da premissa.

Pode-se verificar, que a distinção entre as situações de falsidade e desconhecido pode fornecer informações importantes sobre a inferência em bases de conhecimento e o processo de tomada de decisão. Por exemplo, considere-se um sistema que avalia a participação de determinada pessoa em um crime. Se a resposta obtida pelo sistema é não, [0.0,1.0], pode-se concluir que a pessoa investigada não possui participação e, portanto, é inocente. Porém, uma resposta [0.0,0.0] pode indicar um possível envolvimento e mais informações são necessárias para que uma outra conclusão seja obtida. Em sistemas lógicos convencionais, por exemplo, a linguagem Prolog [Clocksin, 1987] [Sterling, 1986], esta distinção não pode ser obtida diretamente, pois as duas únicas interpretações possíveis são *verdade* ou *não-verdade*. Quando

uma resposta *não-verdade* é obtida, não se sabe se está relacionada à falsidade ou ao desconhecimento do valor-verdade da premissa — *hipótese do mundo fechado* [Rich, 1994].

Diferentemente da Teoria da Probabilidade e outras formas de raciocínio quantitativo, os fatores de crença e descrença não estão diretamente relacionados. Por exemplo, na Teoria da Probabilidade (Seção 3.2), a crença em um determinado evento $A$ é dada por $p(A)$ — probabilidade de ocorrência de $A$ —, logo, a probabilidade de crença em $\neg A$ é dada por $1-p(A)$. Esse e outros relacionamentos não são válidos em Programas Lógicos Evidenciais — PLEs.

A seguir são apresentadas algumas definições lógicas utilizadas na construção e na inferência de Programas Lógicos Evidenciais [Ávila, 1996].

**Definição 4.2.1 (Negação)** O operador de negação $\sim$: $|\mathcal{T}|$ é definido como:

$$\sim ([u, v]) = [v, u]$$

### 4.2.1 Sintaxe

**Definição 4.2.2 (Literal Evidencial)** Se $p$ é uma fórmula básica e

$$u, v \in \{x \in \Re | 0 \leq x \leq 1\}$$

então, diz-se que $p : [u, v]$ é um literal bem anotado e que $[u, v]$ é a anotação de $p$.

**Definição 4.2.3 (Cláusula Evidencial)** Se $p_0 : [u_0, v_0], ..., p_n : [u_n, v_n]$ são literais bem anotados, então

$$p_0 : [u_0, v_0] \Longleftarrow p_1 : [u_1, v_1] \wedge ... \wedge p_n : [u_n, v_n]$$

chama-se cláusula evidencial ou cláusula-e. $p_0 : [u_0, v_0]$ é a cabeça da cláusula, enquanto $p_1 : [u_1, v_1] \wedge ... \wedge p_n : [u_n, v_n]$ é o corpo da cláusula.

**Definição 4.2.4 (Unificação)** Se $p : [u_1, v_1]$ e $q : [u_2, v_2]$ são literais, então diz-se que $p : [u_1, v_1]$ e $q : [u_2, v_2]$ são unificáveis se $p$ e $q$ são unificáveis.

**Definição 4.2.5 (PLE)** Um $PLE$ é qualquer conjunto finito não-vazio de cláusulas-e.

### 4.2.2 Semântica

A base Herbrand [Nicoletti, 1993] [Casanova, 1987] é o domínio de todas as interpretações. Uma interpretação é uma função $I : B_L \rightarrow \mathcal{T}$, tal que $B_L$ é a base Herbrand e $\mathcal{T}$ é o respectivo reticulado. $I(A)$ é o valor verdade associado por $I$ ao literal $A$.

**Definição 4.2.6** A interpretação $I$ satisfaz um literal anotado $p : [u, v]$ se $I(p) \geq [u, v]$ (denotado $I \models p : [u, v]$).

**Definição 4.2.7** A interpretação $I$ satisfaz uma conjunção $p_1 : [u_1, v_1] \wedge ... \wedge \ p_k : [u_k, v_k]$ se $I \models p_i \ \forall \ (1 \leq i \leq k)$.

**Definição 4.2.8** A interpretação $I$ satisfaz uma cláusula-e $p : [u, v] \Longleftarrow q_1 : [u_1, v_1] \wedge ... \wedge \ q_k : [u_k, v_k]$ sse[1]:

- $I \models \ p : [u, v] \wedge q_1 : [u_1, v_1] \wedge ... \wedge \ q_k : [u_k, v_k]$ ou
- $I \not\models \ q_1 : [u_1, v_1] \wedge ... \wedge \ q_k : [u_k, v_k]$.

**Definição 4.2.9** Diz-se que uma interpretação $I$ satisfaz um $PLE$ $E$ se ela satisfaz todas as cláusulas-e de $E$. Portanto $I$ é um modelo para $E$.

De maneira semelhante à ordenação $\leq$ estabelecida sobre as constantes anotacionais, é estabelecida a ordenação entre as interpretações:

[1] Se e somente se.

$$I_1 \leq I_2 \; sse \; (\forall P \in B_E) \; I_1(p) \leq I_2(p)$$

tal que, $B_E$ é a base Herbrand de $E$.

**Definição 4.2.10** Se $E$ é um $PLE$, define-se $T_E$ como uma aplicação de interpretações Herbrand de $E$ em interpretações Herbrand de $E$, tal que

$T_E = \sup\{[u,v] | p : [u,v] \Longleftarrow q_1 : [u_1,v_1] \wedge ... \wedge \; q_k : [u_k,v_k]$ é a instância básica de uma cláusula-e em $E$ e $I \models q_1 : [u_1,v_1] \wedge \; ... \wedge q_k : [u_k,v_k]\}$.

**Definição 4.2.11** Um modelo $I$ que atribui valores-verdade $[u,v]$ ao átomo $p$, sendo $u+v>1$ é dito sobre-determinado.

**Definição 4.2.12** Um modelo $I$ de um $PLE$ $E$ é correto em relação ao átomo $p \in B_E$ se $I(p) = [u,v]$ e $u+v \leq 1$.

**Definição 4.2.13** Um modelo $I$ do $PLE$ $E$ é correto, se é correto em relação a todo átomo $p \in B_E$.

**Definição 4.2.14** Um modelo $I$ do $PLE$ $E$ diz-se completo em relação ao átomo $p \in B_E$ se $I(p) = [u,v]$ e $u+v \geq 1$.

**Definição 4.2.15** Um modelo $I$ do $PLE$ $E$ é completo, se é completo em relação a todo átomo $p \in B_E$.

**Definição 4.2.16** Diz-se que um $PLE$ $E$ é bem-comportado se as cláusulas-e de $E$ satisfazem a seguinte condição:

se $C_1$ e $C_2$ são cláusulas-e em $E$, sendo suas cabeças $p_1 : [u_1,v_1]$ e $p_2 : [u_2,v_2]$ e $p_1$ e $p_2$ são unificáveis, então

$$max(u_1,u_2) + max(v_1,v_2) < 1$$

### 4.2.3 Semântica Operacional dos PLEs

**Definição 4.2.17** Se $E$ é um $PLE$, $p$ é um átomo na linguagem de $E$ e $[u, v]$ uma anotação, define-se uma árvore e/ou $T(E, p[u, v])$ da seguinte forma:

- a raiz de $T(E, p : [u, v])$ é um nó "ou" rotulado $p$:$[u, v]$;
- se $N$ é um nó "ou", então ele é rotulado por um literal anotado simples;
- cada nó "e" é rotulado por uma cláusula-e em $E$ e por uma substituição $\theta$;
- descendentes de nós "ou" são nós "e" e descendentes de nós "e" são nós "ou";
- se $N$ é um nó "ou" rotulado por $p : [u, v]$ e se $C\theta$ é uma instância de uma cláusula-e $C$ em $E$ da seguinte forma: $p : [u_1, v_1] \Longleftarrow q_1 : [\rho_1, \psi_1] \wedge ... \wedge q_k : [\rho_k, \psi_k]$, tal que $[u_1, v_1] \geq [u, v]$, então existe um descendente $N$ rotulado por $C$ e $\theta$. Um nó "ou" sem descendentes chama-se nó *não-informativo*;
- se N é um nó "e" rotulado por uma cláusula-e $C$ e a substituição $\theta$, então para todo literal anotado $p : [u, v]$ no corpo de $C$, existe um nó "ou" descendente rotulado $p\theta : [u, v]$. Um nó "e" sem descendentes chama-se *nó sucesso*.

Associado a cada nó $N$ da árvore *e/ou* definida anteriormente, existe uma constante anotacional $v(N)$ chamada *valor do nó*, da seguinte forma:

- se $N$ é um nó sucesso rotulado $p : [u, v]$, então $v(N) = [u, v]$;
- se $N$ é um nó não-informativo, então $v(N) = [0, 0]$;
- se $N$ é um nó "ou" que não é não-informativo e seus descendentes são $N_1, ...., N_m$, então $v(N) = sup\{v(N_1), .., v(N_m)\}$;
- se $N$ é um nó "e" não-terminal rotulado pela cláusula-e $p : [u, v] \Longleftarrow q_1 : [\rho_1, \psi_1] \wedge ... \wedge q_k : [\rho_k, \psi_k]$, e se o valor $v(N_i)$ de cada um dos nós descendentes $N_i$ rotulados

$q_i$ é tal que $v(N_i) \geq [u_i, v_i]$ para todo $1 \leq i \leq m$, então $v(N) = [u, v]$, senão $v(N) = [0, 0]$.

A seguir é apresentado um exemplo de um $PLE$ e a árvore e/ou gerada. O exemplo é uma extensão do Exemplo 6.2 apresentado em [Blair, 1988].

**Exemplo 4.2.1** Questionamento $p(b) : [1.0, 0.0]$ sobre a base de conhecimento a seguir.(Figura 4.2)

$p(a) : [1.0, 0.0]$.

$p(X) : [1.0, 0.0] \Longleftarrow q(X) : [0.0, 1.0] \wedge r(X) : [1.0, 0.0]$.

$r(a) : [1.0, 0.0]$.

$r(b) : [1.0, 0.0]$.

$q(a) : [0.0, 1.0]$.

$q(b) : [0.0, 1.0]$.

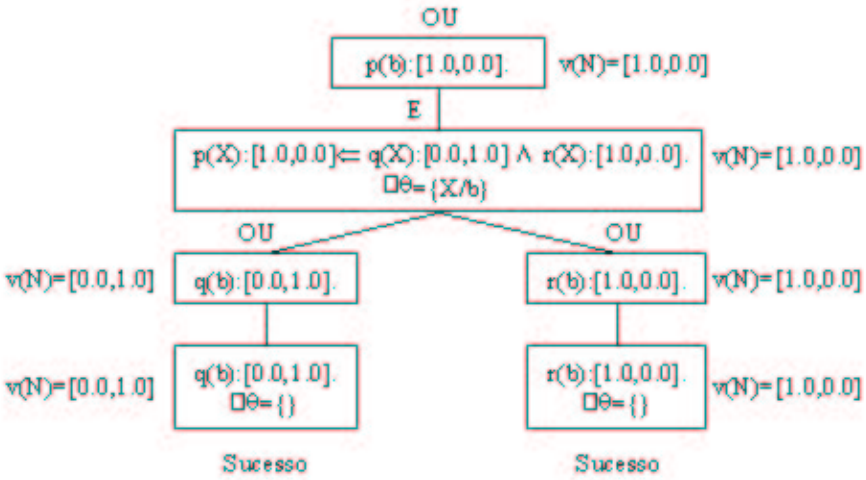



Figura 4.2: Árvore do Exemplo 4.2.1.

A árvore gerada para o Exemplo 4.2.1 é trivial. Isso ocorre porque um $PLE$ que possui apenas anotações perfeitamente definidas — [1.0,0.0] e [0.0,1.0] — e não possui informações conflitantes, produz uma árvore e/ou equivalente à árvore gerada pela Lógica de Primeira Ordem clássica.

Não é possível aplicar diretamente um procedimento de resolução padrão em um $PLE$ [Ávila, 1996]. Dessa forma, foi proposto em [Subrahmanian, 1987b] um procedimento de resolução chamado *resolução-SLDe*. Porém, este procedimento não pode ser aplicado diretamente a um PLE que não seja bem comportado [Ávila, 1996]. Devido à essa restrição, em [Blair, 1988] é proposto um outro procedimento chamado *fechamento*.

**Definição 4.2.18** Um $PLE$ é dito fechado se para quaisquer duas cláusulas-e $C_1$ e $C_2$ pertencentes a $E$ da forma

$$p_1 : [\mu_1, \vartheta_1] \Longleftarrow q_1 : [\rho_1, \psi_1] \wedge ... \wedge q_k : [\rho_k, \psi_k] \; k \geq 0$$
$$p_2 : [\mu_2, \vartheta_2] \Longleftarrow q_1 : [\upsilon_1, \varpi_1] \wedge ... \wedge q_k : [\upsilon_n, \varpi_n] \; n \geq 0$$

tais que $p_1$ e $p_2$ são unificáveis — através de uma unificação mais geral — e $[\mu_1, \vartheta_1]$ e $[\mu_2, \vartheta_2]$ são não-comparáveis, há uma cláusula-e

$$p_1\theta : \mathbf{sup}\{[\mu_1, \vartheta_1], [\mu_2, \vartheta_2] \Longleftarrow q_1 : [\rho_1, \psi_1] \wedge ... \wedge q_k : [\rho_k, \psi_k] \wedge q_1 : [\upsilon_1, \varpi_1] \wedge ... \wedge q_k : [\upsilon_n, \varpi_n]$$

representada por $\lambda(C_1, C_2)$.

**Definição 4.2.19** Seja $E$ um $PLE$ e

$$\begin{aligned} A_1(E) &= E \\ A_{n+1}(E) &= A_n(E) \cup \{\lambda(C_1, C_2) | C_1, C_2 \in A_n(E)\} \; (n \geq 1). \end{aligned}$$

Para todo $PLE$ $E$ há um inteiro $n$ tal que $A_n(E) = A_{n+1}(E)$. $A_n(E)$ chama-se fechamento de $E$ e é denotado $CL(E)$.

**Teorema 4.2.1** Se $E$ é um $PLE$ fechado, então:

1. $I$ é um modelo de $E$ sse $I$ é um modelo de $CL(E)$;

2. $T_E = T_{CL}(E)$.

**Definição 4.2.20** Um $PLE$ sobre $\mathcal{T}$ denomina-se *inconsistente por negação* se há alguma cláusula-e $p : [u, v]$, $p \in B_E$ tal que:

1. $T_E \uparrow w \models p : [u, v]$;

2. $T_E \uparrow w \models p : [v, u]$.

**Definição 4.2.21** Um $PLE$ é dito *não-trivial* se existe alguma cláusula-e $p : [u, v]$ tal que $T_E \uparrow w \not\models p : [u, v]$.

**Definição 4.2.22** Um $PLE$ $E$ é dito *paraconsistente* se $E$ for inconsistente por negação e não-trivial.

### 4.2.4 Grau de Inconsistência e Subdeterminação

O reticulado da Figura 4.1 pode ser transferido para o plano cartesiano (Figura 4.3).

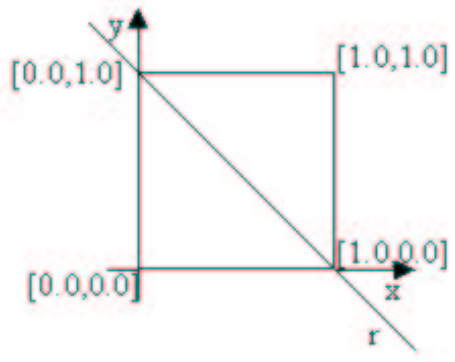



Figura 4.3: Reticulado no Plano Cartesiano.

Pode-se dizer que a reta $r$ é dada pela equação $x + y - 1 = 0$. Portanto, uma cláusula-e $p$:$[u, v]$ é dita perfeitamente definida se $u+v = 1$. De maneira semelhante,

se $u + v < 1$ — é um ponto abaixo da reta $r$ —, a cláusula $p{:}[u, v]$ é chamada subdeterminada. Por outro lado, se $u + v > 1$, $p{:}[u, v]$ é chamada sobredeterminada ou inconsistente.

**Definição 4.2.23** O grau de inconsistência/subdeterminação de $p : [u, v]$ é obtido através de:

$$d(u, v) \times 2^{0.5} \times 100$$

tal que, $d(u, v)$ é a distância do ponto $(u, v)$ em relação à reta $r$ apresentada na Figura 4.3.

## 4.3 Considerações Finais

Nesta Seção foram apresentados os principais conceitos sobre Programação Lógica Evidencial Paraconsistente. Verificou-se que, muitas vezes, informações contraditórias podem ser importantes para o processo de raciocínio e tomada de decisão — como por exemplo, no reconhecimento e na verificação de assinaturas manuscritas — e eliminá-las pode acabar interferindo negativamente na solução de determinado problema, como pode ser verificado no exemplo de diagnóstico médico fornecido na Seção 4.1.

A Programação Lógica Evidencial Paraconsistente fornece um modelo de raciocínio que não elimina a presença de informações contraditórias. Existem infinitos valores-verdade possíveis para uma premissa. Essa característica permite quantificar a inconsistência dos itens de conhecimento envolvidos.

Foi apresentado, através de exemplos, a semântica operacional de PLEs. Porém, como descrito em outros trabalhos [Ávila, 1997] [Ávila, 1998] [Subrahmanian, 1987b] [Blair, 1987], não é possível definir um procedimento de resolução para PLEs que

não sejam bem comportados. Para permitir o raciocínio adequado sobre tais programas, foram apresentadas algumas definições sobre o procedimento de fechamento descrito completamente em [Blair, 1988].

Finalmente, foi introduzido o conceito de grau de inconsistência/subdeterminação, que permite mapear em apenas um valor numérico a inconsistência de uma cláusula-e.

# Capítulo 5

# O Sistema RECSIG

## 5.1 Considerações Iniciais

Pesquisadores de Inteligência Artificial estão envolvidos no desenvolvimento e aprimoramento de métodos computacionais que possam aproximar o comportamento de um computador e aquele de um ser humano. Objetivos ambiciosos como aprendizado, raciocínio, etc., são traçados e alguns problemas podem ser resolvidos. Porém, algumas tarefas triviais para humanos constituem um grande desafio para os pesquisadores. Dentre estas tarefas, pode-se destacar a capacidade de comunicação através da escrita [Gader, 1994] [Gader, 1996].

Computadores capazes de reconhecer a escrita humana seriam muito mais facilmente manipuláveis, e tornariam muito mais amigável sua interface com o usuário. Além disso, diversas tarefas, que hoje são realizadas exclusivamente por humanos, poderiam ser realizadas por computadores de forma muito mais rápida e confiável. Por exemplo, em um ambiente bancário, milhares de cheques são verificados manualmente todos os dias e diversos problemas podem estar relacionados a esta tarefa — incapacidade técnica da pessoa responsável pelo reconhecimento dos campos dos cheques, tempo envolvido para a tarefa, idoneidade do responsável, etc. O reconhe-

cimento de assinaturas manuscritas [Sabourin, 1993] [Sabourin, 1994] [Bastos, 1995] torna-se, portanto, uma etapa fundamental para o processo.

O reconhecimento automatizado de assinaturas manuscritas possui, além do exemplo anterior, inúmeras oportunidades de aplicação, uma vez que qualquer contrato comercial ou processo de identificação pessoal utiliza assinaturas como forma de validação.

Em [Senior, 1992], é ilustrada, de forma interessante, as abordagens empregadas na tarefa de reconhecimento da escrita humana — Figura 5.1.

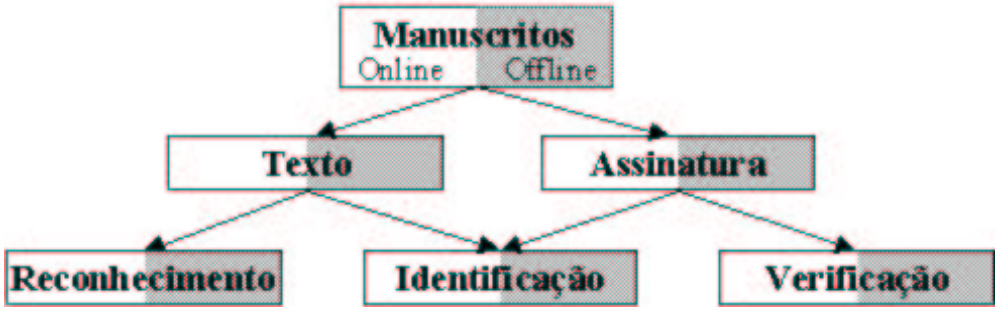



Figura 5.1: Abordagens para Tratamento de Manuscritos.

Basicamente, pode-se perceber através da Figura 5.1, que existem duas estratégias possíveis para um sistema de manipulação de manuscritos — *online* e *offline*.

Um sistema de reconhecimento *online* [Lee, 1996a] utiliza instrumentos acoplados ao computador, como caneta óptica, vídeo sensitivo, etc., para capturar informações. Por outro lado, sistemas *offline* [Lee, 1996] capturam informações a partir de manuscritos já impressos em papel, como por exemplo, sistemas de OCR [Mori, 1992].

Sistemas *online* são mais comuns para a tarefa de reconhecimento manuscrito, uma vez que características importantes para o processo de reconhecimento podem ser obtidas durante a aquisição dos traços, como por exemplo, o tempo utilizado

para obtenção dos traços, quantas vezes a ponta da caneta deixou de estar em contato com a superfície, a força empregada pelo autor, etc. Por outro lado, sistemas *offline* necessitam tratar um grande número de problemas, como por exemplo, tipos diferentes de canetas, sobreposição de traços, tratamento de impurezas na imagem, etc. Porém, verifica-se hoje que uma grande quantidade de documentos impressos continuam sendo verificados e analisados todos os dias. Isto torna necessária a utilização e desenvolvimento de técnicas *offline* de reconhecimento. Outros motivos que impulsionam o reconhecimento *offline* são a constante geração de novos documentos e a não disponibilidade total de instrumentos para aquisição *online* de informações.

Intuitivamente, pode-se dizer que a técnica *offline* é mais semelhante à técnica de reconhecimento humana que a técnica *online*, uma vez que um leitor humano inicia o processo a partir da imagem de todo o manuscrito [Senior, 1992].

Uma outra distinção realizada sobre a técnica adotada em um sistema de manipulação de manuscritos depende do objetivo a ser alcançado pelo sistema. Em sistemas de assinatura manuscrita, dois objetivos podem ser pré-estabelecidos:

- *identificação/reconhecimento*: consiste em identificar o autor de determinada assinatura dentre um conjunto de assinaturas de referência de vários autores;
- *verificação*: consiste em validar a assinatura de um determinado autor, utilizando um conjunto de assinaturas de referência do mesmo autor.

O sistema RECSIG, desenvolvido por Robert Sabourin, Ginette Ginest e Françoise J. Prêteux é um sistema *offline* que realiza a verificação da assinatura manuscrita de um determinado autor através de padrões extraídos localmente. Cada imagem da base de assinaturas consiste de 512×128 pixels. Os padrões são extraídos dividindo a imagem em retinas — pequenas partes da imagem — e aplicando sucessivas operações morfológicas sobre cada retina. A partir dos padrões, o módulo de reconhecimento é acionado e verifica a assinatura do autor.

Nesta Seção, serão apresentados os conceitos de Granulometria utilizados pelo sistema RECSIG [Sabourin, 1997] no seu modelo de aquisição de padrões a partir das imagens. Conceitos básicos de Morfologia Matemática Binária são apresentados no Apêndice C.

## 5.2 Características do RECSIG

Sistemas de reconhecimento de assinaturas buscam elimiar, dentre um conjunto de assinaturas, aquelas que representam falsificações. Segundo Robert Sabourin [Sabourin, 1997], visualmente, é possível identificar falsificações de assinaturas observando as características intrínsecas das assinaturas verdadeiras. Isso pode ser observado considerando a Figura 5.2, onde estão sobrepostas diversas assinaturas de um mesmo autor.

Figura 5.2: Assinaturas Sobrepostas.

Observa-se que as formas, ou características, globais da assinatura permanecem, aumentando o risco de uma assinatura falsa ser classificada como verdadeira pelo sistema. Porém, quando as características individuais do traço de cada autor são quantificadas, o risco de uma classificação errada diminui. Isso pode ser alcançado considerando características que são locais a subregiões da imagem — retinas.

Devido à variabilidade dos traços, a principal dificuldade observada na técnica que utiliza características locais consiste em encontrar características que sejam

adequadas e que estejam relacionadas à identidade do autor. Encontrar, portanto, um descritor de formas que possa extrair tais características da imagem torna-se um desafio.

Geralmente, dois tipos de descritores de forma podem ser encontrados:

- *descritores de forma que preservam informação*: armazenam toda a informação a partir da imagem. Portanto, aplicando o processo inverso à extração de características, é possível recuperar a imagem original;
- *descritores de forma que não preservam informação*: armazenam apenas algumas características da imagem. Portanto, não são capazes de recuperar a imagem inicial.

Especialistas humanos em reconhecimento manuscrito utilizam pequenas peculiaridades para distinguir uma assinatura verdadeira de uma falsa. Descritores de forma que utilizam características locais são classificados como descritores que não preservam informação. Portanto, estes descritores tendem a ser mais adequados para o reconhecimento de assinaturas, enquanto descritores que preservam informação representam características globais da imagem e apresentam desempenho inferior no tratamento de assinaturas [Sabourin, 1997].

A definição do descritor de formas do sistema RECSIG supõe que a área utilizada pelas assinaturas verdadeiras de um autor é relativamente equivalente, ou seja, geralmente um autor tende a utilizar as mesmas retinas para traçar sua assinatura. Além disso, a variabilidade local dos traços é uma característica intrínseca do autor e, portanto, necessita ser quantificada.

O processo de extração de características é iniciado a partir de imagens medindo $512 \times 128$ pixels que são divididas em um certo número de retinas, dependendo das medidas horizontal e vertical de cada retina.

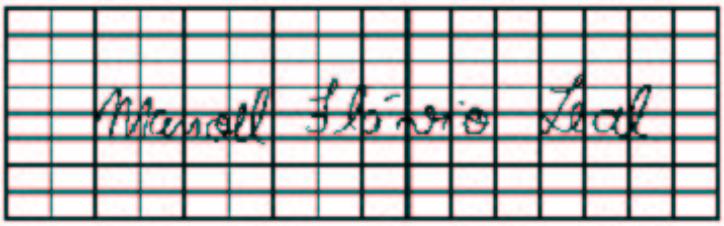


Figura 5.3: Imagem Subdividida em Retinas.

A Figura 5.3 representa uma assinatura dividida em retinas de 32×16 pixels. Para cada retina deve ser obtida uma medida $m$ que representa a excitação da retina.

A variabilidade local dos traços de cada retina, observada na Figura 5.2, é tratada considerando-se, para o processo de extração, não apenas a área da retina, mas um percentual da área das retinas vizinhas verticais e/ou horizontais, dependendo do eixo central da assinatura.

## 5.3 Granulometria e Espectro de Padrões

Técnicas granulométricas, desenvolvidas inicialmente por Matheron [Matheron, 1975], podem ser utilizadas para medir o tamanho e as formas de objetos presentes em uma imagem binária.

Pode ser verificado no Apêndice C que os operadores morfológicos envolvidos podem separar ou eliminar elementos que possuem tamanho e forma diferentes. Por exemplo, a operação de abertura pode eliminar objetos de tamanho pequeno e não alterar grandes objetos. A medida mais comum utilizada para quantificar o tamanho de um objeto é a área, enquanto que medidas de formas podem ser a inclinação, a circunferência, etc.

A Figura 5.4 (I) apresenta uma imagem com dois objetos quadrados — $Q_1$ e $Q_2$ — de tamanhos diferentes. Considerando $a_j$ a área ocupada pelo objeto $j$, pode-

se dizer que a área inicial ocupada pelos quadrados é $área(I) = A_0 = a_{Q_1} + a_{Q_2}$. Assumindo $B_1$ ser um elemento estruturante quadrado menor que $Q_2$, a aplicação da operação de abertura com o elemento estruturante $B_1$ não produz nenhuma alteração na imagem.

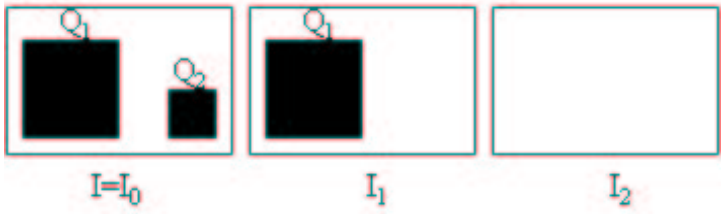



Figura 5.4: Granulometria.

Criando $B_2$ a partir do aumento de $B_1$, tornando $B_2$ maior do que $Q_2$ e menor que $Q_1$, a aplicação da operação de abertura sobre a Figura 5.4 ($I_0$) produz a Figura 5.4 $I_1$, que representa o desaparecimento de $Q_2$. Portanto, a partir de agora, $A_1 = a_{Q_1}$. Criando $B_3$, a partir do aumento de $B_2$, tornando $B_3$ maior do que $Q_1$, através da operação de abertura sobre a Figura 5.4 $I_1$ pelo elemento estruturante $B_3$, é obtida a Figura 5.4 $I_2$, que representa o desaparecimento de $Q_1$. Portanto, a área ocupada na Figura 5.4 $I_2$ — $A_2$ — é zero e não é necessário realizar mais nenhuma iteração.

Pode-se relacionar, portanto, as áreas, obtidas com as operações de abertura realizadas no processo anteriormente descrito, com o tamanho do elemento estruturante. Esta relação está representada na Figura 5.5.

É possível deduzir uma equação que representa a distribuição de probabilidade da área de cada operação de abertura realizada:

$$d(B_n) = \frac{área(I) - área(I\ abe\ B_{(n+1)})}{área(I)} \tag{5.1}$$

tal que, $área(I\ abe\ B_{(n+1)})$ é a área ocupada pelos objetos presentes na imagem obtida a partir da aplicação da operação de abertura na imagem $I$ pelo ES $B_{(n+1)}$.

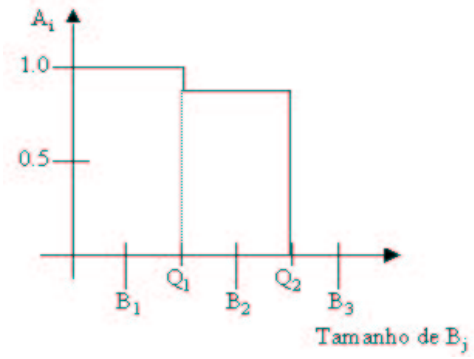



Figura 5.5: Relação entre A e B.

A equação 5.1 é equivalente à proporção da área de $I$ obtida através da operação de abertura. Ela também representa a distribuição granulométrica de tamanho. De forma semelhante à equação 5.1, pode ser definida uma medida de perda, que representa a fração da área inicial que é rejeitada quando uma operação de abertura é realizada com o elemento estruturante $B_{(n+1)}$. Assumindo que a abertura $B_{(n)}$ já foi executada:

$$p(n) = \frac{área(I\ abe\ B_{(n)}) - área(I\ abe\ B_{(n+1)})}{área(I)} \tag{5.2}$$

Na primeira iteração do exemplo apresentado, verifica-se uma situação onde nenhuma informação é adquirida, pois a operação morfológica não alterou a imagem original. Na prática, esta situação tende a ser evitada e todas as iterações do processo granulométrico necessitam fornecer informações. No sistema RECSIG, portanto, a equação completa para o cálculo das áreas perdidas — também conhecida como *pattern spectrum* ou *pecstrum* — é dada da seguinte maneira:

$$p(n) = \frac{área(I\ abe\ B_{(n)}) - área(I\ abe\ B_{(n+1)})}{área(I)},\ n = 0, ..., k-1 \tag{5.3}$$

tal que,

$$\sum_{n=0}^{k} p(n) = 1 \tag{5.4}$$

O *pattern spectrum* para o exemplo anterior é apresentado na Figura 5.6.

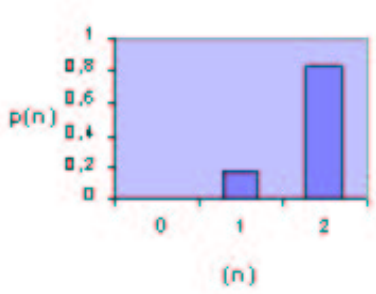



Figura 5.6: Espectro de Padrões.

Obviamente, uma iteração $p(n) = 1$ significa que o objeto desapareceu completamente em apenas 1 passo.

No sistema RECSIG, além da utilização do pecstrum como descritor de formas, foi utilizado o conceito de parte negativa do pecstrum, calculado a partir de:

$$p(-n) = \frac{área(I\ fec\ B_{(n)}) - área(I\ fec\ B_{(n-1)})}{área(I)},\ n = 1, ..., k \tag{5.5}$$

Segundo Sabourin, a parte negativa do *pecstrum* está relacionada a alguma medida normalizada do tamanho das aberturas e cavidades de um objeto. Porém, diferentemente da abordagem vista anteriormente, o *pecstrum negativo* apresenta a desvantagem de não se conhecer o valor máximo para $k$, uma vez que o resultado de sucessivos fechamentos varia, dependendo da forma do ES utilizado.

Um outro descritor utilizado pelo sistema RECSIG é o *pseudopecstrum* — $p_s$. O *pseudopecstrum* foi introduzido por Anastassopoulos [Anastassopoulos, 1991] com o objetivo de superar as dificuldades do *pecstrum negativo* [Sabourin, 1997]. O pseudopecstrum de um objeto $X$ é o pecstrum positivo do complemento $X^c$, relativo

ao círculo mínimo que tem seu centro igual ao centro de gravidade de $X$ e contém $X$. Os valores $p_s(n)$ são normalizados usando a área original de $X$ e a forma do ES é assumida ser circular. O processo é encerrado quando a área de $X^c$ torna-se 0. O resultado obtido pelo *pseudopecstrum* é diferente do obtido pelo *pecstrum negativo*, uma vez que não apenas as cavidades e curvaturas de $X$ são medidas, mas, também se o objeto é similar a um círculo [Sabourin, 1997]. Portanto, pode-se dizer que o *pseudopecstrum* mede o tamanho e a forma de $X$, dependendo do ES.

## 5.4 Representação de Assinaturas Através de Retinas

Como introduzido nas Seções anteriores, a representação de uma assinatura é constituída por um conjunto de regiões — retinas — que possuem, ou não, excitação de pixels. A Figura 5.7 representa uma única retina e seus elementos. Cada $X_i$ representa um objeto presente na retina. A área da retina é delimitada por $W$, enquanto $E$ representa a área total considerada para o tratamento de variabilidade do traço.

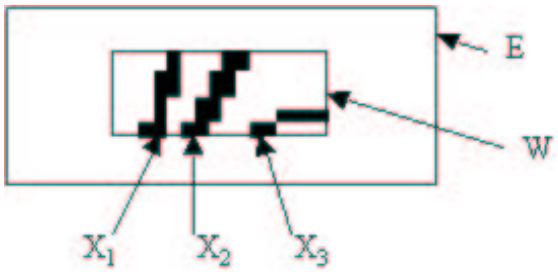



Figura 5.7: Elementos de uma Retina.

Várias são as configurações possíveis para o tratamento das retinas. Por exemplo, pecstrum positivo e pseudopectrum podem ser aplicados com elementos estruturantes {|,/,\,—} e •. Quando o pseudopecstrum baseado em um ES ∈ {|,/,\,—} é

| $h \times v$ | $n_h$ | $n_v$ | $n_r$ | $n$ |
|---|---|---|---|---|
| $16 \times 16$ | 32 | 8 | 256 | 768 |
| $16 \times 32$ | 32 | 4 | 128 | 384 |
| $32 \times 16$ | 16 | 8 | 128 | 384 |
| $32 \times 32$ | 16 | 4 | 164 | 192 |
| $64 \times 32$ | 8 | 4 | 32 | 96 |

Tabela 5.1: Relação Entre o Tamanho e o Número de Retinas de uma Assinatura.

aplicado à $X^c$, o descritor é chamado pseudopecstrum aumentado, pois são considerados percentuais adicionais à área das retinas de 50% na vertical e na horizontal. Porém, quando o ES é circular, nenhum percentual é adicionado. Essa situação é ilustrada na Figura 5.8.


Figura 5.8: Pseudopecstrum.

O número de retinas de uma assinatura está relacionado diretamente ao tamanho das retinas. Considerando que as imagens possuem 512×128 pixels, a Tabela 5.1 apresenta os devidos relacionamentos.

A primeira coluna da Tabela 5.1 representa o tamanho de cada retina, medida pelo número de pixels horizontais e verticais. $n_h$, $n_v$, $n_r$ representam o número de retinas na horizontal, o número de retinas na vertical e o número total de retinas de cada assinatura, respectivamente. O vetor de características, portanto, é obtido por $n_r \times C$, tal que $C$ é o número de características locais para cada retina. No sistema

RECSIG, o número de características é igual a 3, portanto, o tamanho do vetor de características do sistema está descrito na coluna $n$ da Tabela 5.1.

As características utilizadas pelo sistema são:

- *média*: $\mu = \frac{\sum_{n=0}^{k-1} n.p(n)}{área(X)}$;
- *variância*: $\sigma^2 = \frac{\sum_{n=0}^{k-1} (n-\mu)^2.p(n)}{(área(X)-1)}$;
- *inclinação*: $\alpha = \frac{\sum_{n=0}^{k-1} (n-\mu)^3.p(n)}{(\sigma^3.área(X))}$.

Como é possível constatar na Tabela 5.1, o tamanho do vetor de características pode ser muito grande — 768. Porém, geralmente, sistemas de reconhecimento de assinaturas possuem poucas assinaturas de referência. Além disso, há um grande número de retinas que não possuem excitação, e, portanto, serão equivalentes dentro do espaço de características. Estes fatos tornam possível a utilização de algoritmos de Aprendizado de Máquina e Reconhecimento de Padrões para tarefas de reconhecimento e verificação.

## 5.5 Considerações Finais

Nesta Seção foi apresentado o processo de extração de características do sistema RECSIG.

No Apêndice C são apresentados, de maneira sucinta, os principais operadores morfológicos — erosão e dilatação. Pôde-se verificar que cada um deles possui efeitos distintos quando são aplicados sobre uma imagem. Por exemplo, a erosão tende a diminuir os objetos de uma imagem, aumentar possíveis buracos, etc. Por outro lado, a dilatação tende a aumentar os objetos da imagem, preencher buracos, etc. Verificou-se, ainda, que ambas as operações deformam o tamanho da imagem — erosão diminui e dilatação aumenta.

Operações mais completas que a erosão e dilatação também foram apresentadas — abertura e fechamento — de forma que o tamanho dos objetos permanecessem inalterados. Aproveitando-se dessas características, foram apresentados conceitos de granulometria.

Os conceitos de granulometria foram empregados na apresentação de descritores de forma definidos no sistema RECSIG [Sabourin, 1997]. Tal sistema, desenvolvido por Robert Sabourin, Ginette Genest e Françoise J. Prêteux, utiliza distribuições granulométricas de tamanho para quantificar formas e medidas de excitação de retinas de imagens de assinaturas manuscritas. O sistema propõe quantificar as características intrínsecas da assinatura de cada autor, e, posteriormente, utilizar os valores obtidos como forma de distinguir a identidade individual de cada um. As características são consideradas localmente, ou seja, as imagens são divididas em subregiões — retinas — e as características extraídas a partir de cada retina. No presente trabalho, foi enfocado apenas o processo de quantificação das formas do sistema RECSIG.

Em [Sabourin, 1997] estão descritos os resultados obtidos pelo RECSIG a partir de uma base de imagens de assinaturas manuscritas. Os resultados demonstram altas taxas de precisão do sistema. Esses resultados serão utilizados como base de comparação com o método de aprendizado que este trabalho pretende produzir e deverão ser apresentados em Seções posteriores.

# Capítulo 6

# PrLE Paraconsistente em Árvore de Decisão

## 6.1 Considerações Iniciais

Como visto anteriormente — Capítulo 2 —, AM consiste em obter conceitos — conhecimento — a partir de bases de dados. O conhecimento extraído a partir dos dados pode ser utilizado para definir algum comportamento inteligente em um determinado ambiente.

Um dos paradigmas mais conhecidos de AM é o que utiliza indução através de AD. Dentre os algoritmos mais conhecidos de AD estão o ID3 [Quinlan, 1986], o C4.5 [Quinlan, 1993] e o C5.0 [Carvalho, 1999]. Esses algoritmos são baseados no algoritmo de Hunt [Hunt, 1966], e são empregados satisfatoriamente em muitas aplicações nas mais variadas áreas. Porém, todos esses algoritmos apresentam anomalias quando são empregados sobre bases de dados que possuem informações inconsistentes.

Uma base de dados é dita inconsistente, quando possui exemplos de treinamento

cujos valores de seus atributos previsores são idênticos, porém, os valores do atributo-meta são diferentes.

Geralmente, a execução de algoritmos de AM sobre bases de dados inconsistentes tende a gerar conceitos inconsistentes. Por exemplo, observando as seguintes regras geradas a partir de um algoritmo de aprendizado qualquer sobre uma base de dados BD:

$$\textbf{se } a_1 = 1 \textbf{ então } classe = C$$

$$\textbf{se } a_1 = 1 \textbf{ e } a_2 = 2 \textbf{ então } classe = D$$

tal que, $a_i$ é um atributo previsor e *classe* um atributo meta. Pode-se dizer, que há uma inconsistência entre as regras. Segundo Beat Wüthrich [Wüthrich, 1997], regras são ditas inconsistentes quando cobrem os mesmos exemplos — no exemplo anterior, observações que possuem $a_1 = 1$ —, porém, os classificam como pertencentes a classes diferentes. Esse fenômeno também pode ser observado quando conceitos, obtidos a partir de fontes de dados diferentes, são unidos para formar apenas uma base de conhecimento [Ávila, 1996] [Benferhat, 1995].

O problema de exemplos de treinamento inconsistentes pode ser abordado, basicamente, de duas formas:

1. atribuir ao algoritmo de aprendizado a habilidade de manipular adequadamente as informações conflitantes durante o processo de aprendizado, gerando, dessa forma, conceitos consistentes e confiáveis;

2. aplicar sobre o conhecimento obtido, através de algum algoritmo de aprendizado qualquer, um método de raciocínio que possibilite a inferência confiável de informações: transformações na base de conhecimento de modo a torná-la consistente, geração de outra base de conhecimento a partir dos itens consistentes de conhecimento existentes na base inicial, etc.

No segundo caso, podem ser utilizados métodos de racicíonio quantitativo, como os apresentados em [Subrahmanian, 1987a] [Subrahmanian, 1987b] [van Emden, 1986]. Além disso, técnicas de Gerenciamento de Incerteza (Capítulo 3) [Zhou, 1995] como Raciocínio Probabilístico, Lógica Fuzzy, Fatores de Certeza, etc., também poderiam ser utilizados.

Neste trabalho, pretende-se abordar o problema apresentado estendendo ao algoritmo básico de AD a habilidade de manipular exemplos de treinamento inconsistentes — abordagem 1. Para tanto, conceitos de Raciocínio Quantitativo, Gerenciamento de Incerteza e Lógica Paraconsistente serão empregados com o objetivo de medir individualmente a inconsistência de cada exemplo de treinamento, além de quantificar o grau de inconsistência de um subconjunto de exemplos de treinamento. Pode-se, portanto, medir a qualidade de determinados conceitos, em termos de consistência, de acordo com o grau de consistência obtido pelos respectivos subconjuntos de exemplos cobertos. Dessa forma, comparações relacionadas à qualidade dos conceitos gerados, em termos de classificação, podem ser realizadas entre o método proposto e os métodos tradicionais de aprendizado, através de árvores de decisão e de outros paradigmas.

## 6.2 Aprendizado e Programação Lógica Evidencial

Como introduzido anteriormente — Capítulo 4 — pode-se utilizar mecanismos fornecidos pela Lógica Paraconsistente para manipulação adequada de informações inconsistentes. Nesta Seção são empregados conceitos de Lógica Paraconsistente e Árvore de Decisão na definição de um modelo de Aprendizado de Máquina capaz de realizar inferências sobre bases de dados que possuem informações conflitantes [Enembreck, 1999].

Para que conceitos de Lógica Paraconsistente possam ser utilizados em AM, elementos que formam um sistema de aprendizado necessitam ser definidos logicamente. A seguir serão apresentadas várias definições lógicas de tais elementos.

Como consta no Apêndice A.2, um exemplo de treinamento $e_i$ é formado por um conjunto de valores $< v_1, ..., v_m >$, tal que cada $v_i$ representa um valor possível para o atributo $a_i$. Os $n = m - 1$ atributos $\{a_1, ..., a_n\}$ representam os atributos previsores e o atributo $a_m$ é o atributo meta. Cada $v_i \in d_i$, tal que $d_i$ representa o conjunto de valores possíveis para o atributo $a_i$, ou seja, o domínio de $a_i$.

A partir das definições anteriores, podem ser definidos símbolos básicos de uma linguagem $L$:

1. variáveis individuais: x, y, z, ...;
2. se $v \in \{d_1 \cup d_2 \cup ... \cup d_n\}$ então $v$ é um *termo*;
3. se $p \in d_m$ então $p(v_1,...,v_n)$ existe, e $p$ é um símbolo predicativo $n$-ário;
4. se $p$ é um símbolo predicativo $n$-ário, e $v_i$ é um termo, então $p(v_1,...,v_n)$ é uma fórmula básica;
5. se $\mu_1$ e $\mu_2 \in \{x \in \Re | 0 \leq x \leq 1\}$ então $\mu_1$ e $\mu_2$ são constantes anotacionais;
6. se $A$ é uma fórmula básica e $\mu_1$,$\mu_2$ são constantes anotacionais, então $A$:$[\mu_1,\mu_2]$ é um literal bem-anotado e $[\mu_1,\mu_2]$ é uma anotação de $A$;
7. se $E$ é um conjunto não-vazio de literais bem-anotados, então $E$ é um *Programa Lógico Evidencial*;
8. variável anônima: "_".

É importante ressaltar a extrema simplicidade da linguagem definida. Isso se deve ao fato que um conjunto de treinamento representa um conjunto de fórmulas

básicas e fatos lógicos. Além disso, não existem estruturas da forma $A \longleftarrow B$, tal que $A$ representa uma fórmula básica (cabeça) e $B$ representa uma conjunção ou disjunção de fórmulas (corpo). Portanto, é natural que o mecanismo de raciocínio também seja muito mais simples que os definidos por van Endem [van Emden, 1986] e Blair [Blair, 1987].

Como definido na Seção 4.2, $\mathcal{T} =< |\mathcal{T}|, \leq >$, tal que $|\mathcal{T}| = \{x \in \Re | 0 \leq x \leq 1\} \times \{x \in \Re | 0 \leq x \leq 1\}$. Pode-se considerar, portanto, que o conjunto formado por todas as fórmulas básicas de um Programa Lógico Evidencial $E$, constitui a base de Herbrand de $E$ [Nicoletti, 1993] [Casanova, 1987], denotada por $B_E$. Uma interpretação é uma função $I{:}B_E \rightarrow \mathcal{T}$, tal que $B_E$ é a base Herbrand em consideração e $\mathcal{T}$ é o reticulado subjacente.

**Definição 6.2.1** A interpretação $I$ satisfaz um literal anotado $p : [u, v]$ se $I(p) \geq [u, v]$ (denotado $I \models p : [u, v]$).

**Definição 6.2.2** A interpretação $I$ satisfaz o Programa Lógico Evidencial $E$ se satisfaz todas as cláusulas de $E$.

**Definição 6.2.3** $\mathrm{I}_1 \leq I_2$ *sse* $(\forall p \in B_E)\ I_1(p) \leq I_2(p)$, tal que $E$ é o $PLE$ e $B_E$ é a base de Herbrand de $E$.

**Definição 6.2.4 (Unificação de Termos)** Se $p$ e $q$ são termos, então $p$ e $q$ são unificáveis sse: $p$ e $q$ são iguais ou $p$ ou $q$ são variáveis anônimas.

**Definição 6.2.5 (Unificação de Fórmulas Básicas)** Se $p(t_1, ..., t_n)$ e $q(s_1, ..., s_n)$ são fórmulas, então são unificáveis sse $p$ e $q$ são unificáveis e $\forall i \in \{1, ..., n\}$ $t_i$ e $s_i$ são unificáveis.

**Definição 6.2.6 (Unificação de Literais)** Se $A : [u_1, v_1]$ e $B : [u_2, v_2]$ são literais, então são unificáveis sse $A$ e $B$ são unificáveis.

**Definição 6.2.7** Se $E$ é um $PLE$, define-se $T_E$ como uma aplicação de interpretações Herbrand de $E$ em interpretações Herbrand de $E$, tal que $T_E(I)(p) = sup\{[u,v]|p : [u,v]$ é a instância básica de uma cláusula em $E\}$.

### 6.2.1 Semântica Operacional Definida para PLEs

Diferentemente dos trabalhos de Ávila [Ávila, 1996], Blair [Blair, 1987] e Subrahmanian [Subrahmanian, 1987b], não é necessário construir uma árvore e/ou no presente sistema. Isso se deve ao fato que na linguagem definida, os literais não possuem corpo, portanto, não há nós "$e$" para serem colocados em uma possível árvore de resolução. Além disso, a árvore seria formada apenas pela raiz e o conjunto de literais unificados.

**Definição 6.2.8 (Nó Não-Informativo)** Um nó "ou" sem descendentes é chamado de nó não-informativo.

A árvore $T$ gerada pelo sistema é, portanto, formada apenas por nós *"ou"* e é representada da seguinte forma:

$$T(E, p : [\mu_1, \mu_2]).$$

Associada a cada nó $N$ da árvore existe uma constante anotacional *v(N)* chamada valor do nó.

**Definição 6.2.9** Se $N$ é um nó não-informativo, então $v(N) = [0, 0]$.

**Definição 6.2.10** Se $N$ é um nó que não é não-informativo, ou seja, possui descendentes $\{N_1, ..., N_m\}$, então $v(N) = sup\{N_1, ..., N_m\}$.

O procedimento de resolução *SLD* [Casanova, 1987] não pode ser aplicado diretamente a um *PLE* [Ávila, 1996]. Para *PLEs*, um procedimento chamado fechamento — Seção 4.2.3, página 52 — é descrito completamente em [Blair, 1988].

A seguir é fornecido um exemplo de um $PLE$ e a sua árvore de resolução equivalente.

**Exemplo 6.2.1** Dada a seguinte base de conhecimento, a árvore gerada pelo sistema para o questionamento $p : [0, 0]$ é ilustrada na Figura 6.1.

p:[0.80,0.30].

p:[0.20,0.60].

p:[0.90,0.10].

p:[0.80,0.70].

q:[0.50,0.50].

q:[0.78,0.92].


Figura 6.1: Árvore OU Gerada.

### 6.2.2 Semântica Operacional de PLEs em Árvore de Decisão

Como definido nas Seções anteriores, uma árvore de decisão é formada por um conjunto de nós $N$ e um conjunto de arcos que ligam estes nós. Um nó $N_i \in N$ representa um subconjunto $E_{N_i}$ do conjunto total de exemplos de treinamento $E$. Se $N_i$ é raiz, então $E_{N_i} \subseteq E$ e $E \subseteq E_{N_i}$. Nós folha são denotados $N_F$.

| | | | |
|---|---|---|---|
| 1 | $classe_1(2,4,6):[u_1, v_1].$ | 5 | $classe_2(1,3,8):[u_5, v_5].$ |
| 2 | $classe_1(2,4,7):[u_2, v_2].$ | 6 | $classe_2(1,3,6):[u_6, v_6].$ |
| 3 | $classe_1(2,4,8):[u_3, v_3].$ | 7 | $classe_2(1,4,7):[u_7, v_7].$ |
| 4 | $classe_1(1,4,7):[u_4, v_4].$ | 8 | $classe_2(1,3,6):[u_8, v_8].$ |

| | |
|---|---|
| 9 | $classe_3(1,3,6):[u_9, v_9].$ |
| 10 | $classe_3(2,3,7):[u_{10}, v_{10}].$ |
| 11 | $classe_3(2,5,8):[u_{11}, v_{11}].$ |
| 12 | $classe_3(2,5,6):[u_{12}, v_{12}].$ |

Tabela 6.1: Programa $PLE\ E$.

O mecanismo de inferência, obviamente, satisfaz estruturas que pertencem à linguagem $L$ definida na Seção 6.2, página 72. Dessa forma, cada exemplo do sistema de aprendizado representa um literal bem-anotado. A seguir é apresentado um exemplo correspondente a um questionamento.

**Exemplo 6.2.2** Seja uma linguagem $U$ definida por um conjunto de exemplos, tal que cada exemplo é formado por um conjunto $A = \{a_1, ..., a_4\}$ de atributos. O domínio $d_i$ de cada atributo $a_i$ é definido da seguinte forma:

$$d_1 = \{1, 2\} \quad d_2 = \{3, 4, 5\} \quad d_3 = \{6, 7, 8\} \quad d_4 = \{classe_1, classe_2, classe_3\}$$

tal que, $d_1,\ d_2,\ d_3$ são atributos previsores e $d_4$ é o atributo meta.

Seguindo definições anteriores, o $PLE\ E$ é dado na Tabela 6.1.

Uma possível árvore de decisão gerada a partir dos exemplos fornecidos é apresentada na Figura 6.2.

Dado um questionamento $Q = classe_2(1, 3, 7) : X$, tal que $X$ é uma variável anotacional, o procedimento percorre a árvore de decisão, a partir da raiz, realizando os devidos testes, até encontrar um nó-folha. No exemplo, o nó-folha encontrado é $N_4$. $N_4$ é considerado um $PLE$ individual, portanto, a semântica descrita na Seção 6.2.1 é mantida. Uma operação de modificação $M$ é realizada no questionamento. A


Figura 6.2: Árvore de Decisão Gerada.

modificação consiste em substituir todos os termos que não são utilizados pela AD por variáveis anônimas, dessa forma,

$$M(Q) = Q' = classe_2(1, 3, \_) : X$$

uma vez que, para esse exemplo, apenas o primeiro e o segundo atributo são utilizados pela árvore para classificação.

Essa etapa é necessária para que o fechamento e o cálculo do supremo (Seção 4.2.3) possam ser realizados corretamente. Finalmente, o resultado é obtido:

$$X = \sup\{[u_5, v_5], [u_6, v_6], [u_8, v_8]\}$$

É importante notar que o exemplo de treinamento 9 pertencente à $N_4$ não é utilizado durante a inferência realizada para o questionamento $Q'$. Isto se deve ao fato de que a unificação entre $Q'$ e o exemplo 9 não é possível.

## 6.3 Utilização do Modelo de Árvore de Decisão Paraconsistente para a Tarefa de Classificação

Uma das principais finalidades de uma Árvore de Decisão é obter a maior taxa de precisão para a classificação de exemplos que não foram utilizados para o treinamento, ou seja, exemplos não conhecidos. Quando esse objetivo é alcançado, pode-se

dizer que os conceitos gerados estão consistentes em relação ao modelo e o sistema obteve um modelo correto a partir dos dados.

No modelo de Árvore de Decisão Paraconsistente, existem duas formas de questionamento [Enembreck, 1999]:

1. o questionamento é realizado fornecendo como entrada os valores de todos os atributos, inclusive o valor do atributo-meta. Neste caso, a resposta retornada será o fator evidencial calculado a partir do supremo do respectivo nó-folha — Exemplo 6.2.2.

2. o questionamento é realizado informando-se apenas os valores dos atributos previsores. Neste caso, o sistema necessita escolher uma dentre as classes possivelmente existentes no nó-folha correspondente.

Na segunda forma de questionamento, uma questão importante necessita ser resolvida: *Qual o critério para a escolha de uma determinada classe em uma folha?* Para justificar o critério de escolha adotado é fornecido o Exemplo 6.3.1.

**Exemplo 6.3.1** Seja $Q_e$ um questionamento com o seguinte significado: *Qual a classe do exemplo de teste* $\boldsymbol{e}$*?* O exemplo $e$ é da forma $e = classe(a_1, ..., a_n)$, tal que $a_i$ é um atributo-previsor e *classe* é a classe do exemplo. É assumido, também, que a travessia na AD conduziu o mecanismo de inferência para o nó-folha $N_F$, formado pelo subconjunto $E_F \subseteq E$ de exemplos — Tabela 6.2 —, tal que $E$ é o conjunto de todos os exemplos de treinamento.

Obviamente, quando uma folha $NF$, é formada por um conjunto $E_F \subseteq E$ de exemplos, que possuem apenas uma determinada classe $C$, a resposta ao questionamento $Q$ é dada por:

$$Q = C : \sup\{E_F\} \tag{6.1}$$

| |
|---|
| $classe_1(e) : [0.50, 0.12].$ |
| $classe_1(e) : [0.62, 0.07].$ |
| $classe_2(e) : [0.02, 0.29].$ |
| $classe_2(e) : [0.18, 0.90].$ |

Tabela 6.2: Nó Folha $N_F$.

tal que, $C : sup\{E_F\}$ representa o supremo obtido a partir do subconjunto de exemplos de $E_F$ que possuem classe $C$.

Porém, no Exemplo 6.3.1 — Tabela 6.2 —, o nó-folha $N_F$ apresenta exemplos de duas classes — $C_1 = classe_1$ e $C_2 = classe_2$. Neste caso, é necessário calcular para cada classe $C_i$, o supremo dos exemplos pertencentes a ela — conjunto $E_{F_i}$ —, denotado $sup\{E_{F_i}\}$. Os supremos de $C_1$ e $C_2$ são dados por:

- $C_1$: $\sup\{E_{F_1}\} = [0.62, 0.12]$;
- $C_2$: $\sup\{E_{F_2}\} = [0.18, 0.90]$.

Intuitivamente, a partir dos supremos obtidos, poder-se-ia calcular seus respectivos fatores de inconsistência/subdeterminação (Seção 4.2.4) e escolher dentre as classes aquela que apresentasse o menor fator. No Exemplo 6.3.1, isso significaria escolher a classe $C_2$, uma vez que:

$$IS(\sup\{E_{F_1}\}) = 0.26 > IS(\sup\{E_{F_2}\}) = 0.08$$

Porém, observando mais cuidadosamente os valores evidenciais dos supremos das classes $C_1$ e $C_2$, verifica-se um valor muito mais favorável à crença em $C_1$ do que a crença em $C_2$. Além disso, os valores evidenciais do supremo de $C_2$ mostram uma descrença muito maior do que a crença no próprio $C_2$. Portanto, para o supremo de cada classe $C_i$ é calculado um fator de certeza $FC_i$ (Seção 3.4) e escolhida a classe que apresenta o maior fator de certeza. No exemplo, isso significa que a classe $C_1$ é escolhida, uma vez que:

$$FC_1(\sup\{E_{F_1}\}) = 0.5 > FC_2(\sup\{E_{F_2}\}) = -0.72$$

O fator de inconsistência/subdeterminação atribui, por exemplo, às evidências $e_1 = [0.8, 0.0]$ e $e_2 = [0.0, 0.8]$, a mesma importância, ou seja, possui uma característica simétrica. Porém, $e_1$ possui claramente uma melhor qualidade, ou seja, maior evidência de ser a hipótese correta. Através do critério baseado no fator de certeza, pode-se identificar que $e_1$ é a melhor hipótese.

O segundo critério — baseado no fator de certeza — é o utilizado pelo sistema, uma vez que a característica simétrica do fator de inconsistência/subdeterminação não é desejável. Portanto, pode-se concluir que não há relação direta entre a inconsistência de determinada classe e seu grau de pertinência em relação a um dado exemplo.

## 6.4 Comparação Entre ADP e C4.5

Com o objetivo de comparar o comportamento dos algoritmos ADP e C4.5 foi utilizado um procedimento iterativo com 10 iterações sobre um conjunto de base de dados. Foi estipulado, arbitrariamente, que 70 % dos exemplos seriam utilizados para treinamento e 30 % para teste. As bases utilizadas nos experimentos são as seguintes: Myoeletric, Lenses, Hayes, Zoo, Iris, Esporte, Wine, Glass, Flag, Adult, Adult1.

A descrição detalhada de cada base utilizada nos experimentos pode ser obtida em [Murphy, 1995], exceto a base *Esporte*, apresentada de forma detalhada em [Quinlan, 1993] [Mitchell, 1997]. Para cada base, vários classificadores — ADPs — foram obtidos, uma vez que diversos fatores de poda foram testados. A partir desse conjunto de classificadores, para cada base, foi selecionado aquele que apresentou a melhor taxa de classificação nos dados de teste. Além disso, um outro parâmetro adicional foi utilizado, e representa a percentagem de exemplos inconsistentes que

devem ser gerados automaticamente antes do treinamento.

A seguir é descrito o procedimento utilizado para geração automática de exemplos inconsistentes.

Dado que $E$ é o conjunto de exemplos de uma determinada base, $C$ é o conjunto de classes e $P \in [0, 1]$ o percentual de inconsistência fornecido como parâmetro, a geração dos exemplos inconsistentes é dada da seguinte forma:

- repetir até que a percentagem $P$ de exemplos inconsistentes seja alcançada;
  - escolher um exemplo $E_i \in E$ aleatoriamente, escolha uma classe $C_j \in C$ diferente da classe de $E_i$;
  - criar um novo exemplo $E_k$ formado pelo conjunto de valores dos atributos previsores do exemplo $E_i$ e pela classe $C_j$ e armazenar o exemplo $E_k$ em uma base temporária $T$;
- unir as bases $E$ e $T$.

A partir das bases de dados inconsistentes, o sistema foi avaliado em termos de precisão de classificação e tamanho das árvores geradas. As Figuras 6.3 e 6.4 apresentam o erro médio de classificação obtido e o tamanho médio das árvores geradas, respectivamente, pelos algoritmos C4.5 e ADP, à medida em que o fator de inconsistência é aumentado. Esses dados referem-se às últimas colunas das Tabelas 6.3 e 6.4, respectivamente.

Quanto aos resultados de classificação obtidos, pode-se verificar que para bases apresentando poucos exemplos inconsistentes, o comportamento do sistema ADP é muito semelhante ao C4.5. Pode-se observar na Figura 6.3 que para taxas de inconsistência que variam de 0.0 a 0.3, o sistema apresenta resultados próximos ao C4.5. Porém, a partir do fator 0.4 de inconsistência, os resultados apresentam uma sensível melhora na taxa de erro de classificação, comprovando que o sistema é

mais estável quando submetido a bases de dados que apresentam elevadas taxas de inconsistência.

Os resultados apresentados na Figura 6.4 demonstram que para bases de dados que apresentam baixas taxas de inconsistência, as árvores geradas tendem a ser mais complexas em relação àquelas obtidas com o algoritmo C4.5. A geração excessiva de intervalos para determinados atributos, causada pela discretização, pode ser o principal fator responsável pela obtenção de árvores de decisão complexas. Além disso, o critério simples de poda adotado também pode contribuir para o crescimento excessivo das árvores. Porém, a partir do fator 0.5 de inconsistência, observa-se que o sistema tende a obter árvores menores.

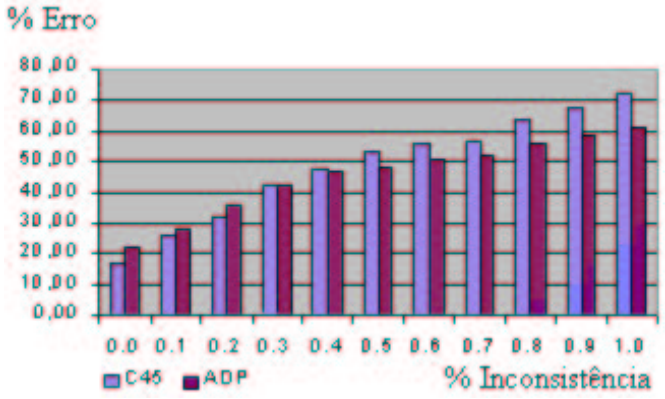



Figura 6.3: Erro Médio de Classificação.

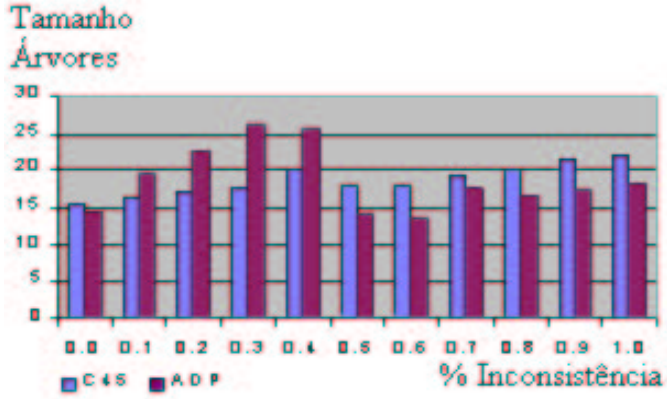



Figura 6.4: Tamanho Médio das Árvores.

| % Inc. | Myo | Lenses | Hayes | Zoo | Iris | Esporte | Wine | Glass | Flag | Adult | Adult1 | Média |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0,0 | 15,09 | 12,15 | 26,87 | 9,37 | 6,95 | 50,80 | 8,48 | 26,60 | 29,35 | 5,00 | 0,00 | 17,33 |
| | 10,99 | 22,14 | 39,20 | 11,70 | 5,82 | 30,00 | 37,78 | 40,20 | 43,90 | 5,00 | 0,00 | 22,43 |
| 0,1 | 22,23 | 20,54 | 34,49 | 15,60 | 13,07 | 59,50 | 17,07 | 40,18 | 40,63 | 17,39 | 5,72 | 26,04 |
| | 21,86 | 21,96 | 41,27 | 32,14 | 13,26 | 30,00 | 26,46 | 46,95 | 50,39 | 10,23 | 13,33 | 27,99 |
| 0,2 | 25,56 | 30,68 | 37,74 | 25,55 | 22,49 | 35,50 | 25,99 | 50,35 | 48,78 | 20,74 | 26,43 | 31,80 |
| | 27,90 | 37,50 | 46,92 | 35,37 | 21,18 | 39,50 | 35,47 | 51,11 | 53,65 | 24,04 | 20,71 | 35,76 |
| 0,3 | 37,50 | 48,46 | 46,87 | 30,25 | 26,33 | 59,00 | 30,62 | 54,45 | 51,97 | 35,19 | 40,54 | 41,93 |
| | 33,00 | 48,33 | 52,47 | 47,94 | 27,09 | 38,50 | 35,43 | 54,65 | 54,89 | 35,00 | 37,50 | 42,25 |
| 0,4 | 47,28 | 46,44 | 45,46 | 36,78 | 31,58 | 57,00 | 37,50 | 62,41 | 60,08 | 52,15 | 39,84 | 46,96 |
| | 42,26 | 51,55 | 53,51 | 46,88 | 32,85 | 46,33 | 42,45 | 58,96 | 61,31 | 40,35 | 38,75 | 46,84 |
| 0,5 | 52,74 | 45,01 | 55,09 | 39,14 | 37,01 | 47,66 | 52,20 | 63,76 | 67,54 | 67,80 | 55,70 | 53,06 |
| | 43,50 | 46,99 | 55,94 | 46,31 | 35,06 | 39,33 | 47,02 | 58,41 | 65,35 | 44,86 | 48,61 | 48,31 |
| 0,6 | 55,15 | 58,44 | 56,69 | 43,20 | 45,31 | 61,20 | 52,86 | 65,31 | 64,42 | 56,88 | 52,56 | 55,64 |
| | 49,73 | 51,00 | 60,67 | 49,15 | 38,46 | 41,42 | 52,40 | 61,70 | 65,90 | 42,88 | 46,55 | 50,90 |
| 0,7 | 61,36 | 68,04 | 21,20 | 46,62 | 46,66 | 61,90 | 57,46 | 71,25 | 70,83 | 65,22 | 50,09 | 56,42 |
| | 49,02 | 54,40 | 27,60 | 54,32 | 44,78 | 48,50 | 52,20 | 65,16 | 68,06 | 52,77 | 50,11 | 51,54 |
| 0,8 | 63,24 | 65,17 | 64,42 | 53,71 | 56,29 | 57,14 | 63,16 | 72,00 | 72,39 | 61,43 | 68,07 | 63,37 |
| | 49,09 | 55,19 | 63,27 | 59,57 | 47,80 | 48,09 | 55,29 | 66,11 | 70,17 | 54,18 | 47,63 | 56,04 |
| 0,9 | 70,02 | 77,25 | 66,34 | 58,43 | 56,15 | 63,37 | 72,87 | 74,18 | 77,45 | 63,70 | 60,99 | 67,34 |
| | 48,30 | 58,51 | 67,01 | 62,24 | 54,60 | 50,89 | 59,10 | 64,91 | 72,16 | 52,81 | 57,36 | 58,90 |
| 1,0 | 71,56 | 72,25 | 69,69 | 63,27 | 63,83 | 72,85 | 74,51 | 81,07 | 79,00 | 70,36 | 71,21 | 71,78 |
| | 48,98 | 71,40 | 66,36 | 66,55 | 54,90 | 50,71 | 58,96 | 71,35 | 73,39 | 54,09 | 56,06 | 61,16 |

C45
ADP

Performance Superior
ADP

Tabela 6.3: Taxas de Erros de Classificação Obtidas.

As Tabelas 6.3 e 6.4 apresentam os resultados de todos os experimentos realizados. Como já citado, o sistema obteve melhores resultados na grande maioria dos casos onde elevadas taxas de exemplos inconsistentes estão presentes. Porm, a Tabela 6.3 apresenta uma característica interessante: para a base de dados Zoo, o sistema não obteve performance superior para nenhum percentual de inconsistência nos dados. Tal base apresenta 16 atributos dos quais 15 são atributos binários. Esta característica pode indicar que a função a ser estimada tende a ficar muito mais complexa à medida em que o percentual de exemplos inconsistentes aumenta. Esse comportamento sugere que o sistema apresenta de forma mais acentuada, em relação à algoritmos convencionais de árvores de decisão, o problema de busca local, implementada pela escolha individual do melhor atributo. Essa hipótese pode ser

| % Inc. | Myo | Lenses | Hayes | Zoo | Iris | Esporte | Wine | Glass | Flag | Adult | Adult1 | Média |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0,0 | 6,40 | 5,80 | 24,50 | 14,60 | 7,00 | 3,60 | 11,60 | 39,60 | 47,60 | 4,80 | 5,00 | 15,50 |
| | 5,55 | 6,44 | 41,66 | 16,77 | 5,00 | 1,00 | 12,22 | 51,50 | 8,56 | 3,80 | 4,00 | 14,23 |
| 0,1 | 8,00 | 6,00 | 24,70 | 18,20 | 6,20 | 2,60 | 12,60 | 42,40 | 48,50 | 5,20 | 4,80 | 16,29 |
| | 5,11 | 6,77 | 46,67 | 22,44 | 8,11 | 1,00 | 32,77 | 53,10 | 29,70 | 5,20 | 4,90 | 19,62 |
| 0,2 | 6,40 | 4,60 | 24,30 | 13,40 | 7,80 | 2,00 | 16,20 | 50,20 | 53,90 | 4,40 | 3,60 | 16,98 |
| | 11,10 | 9,00 | 24,89 | 56,33 | 9,33 | 1,00 | 34,11 | 64,70 | 28,30 | 4,60 | 4,90 | 22,57 |
| 0,3 | 7,80 | 3,60 | 20,60 | 17,40 | 7,60 | 3,70 | 16,20 | 50,60 | 55,60 | 5,40 | 4,40 | 17,54 |
| | 12,22 | 9,67 | 33,44 | 10,33 | 7,67 | 1,00 | 32,77 | 90,70 | 77,40 | 2,20 | 9,50 | 26,08 |
| 0,4 | 7,80 | 1,40 | 22,10 | 17,60 | 7,00 | 1,80 | 37,50 | 57,60 | 61,80 | 2,40 | 4,20 | 20,11 |
| | 9,11 | 5,00 | 21,22 | 13,11 | 7,60 | 1,00 | 42,45 | 92,60 | 87,00 | 1,00 | 2,00 | 25,64 |
| 0,5 | 8,40 | 4,40 | 19,40 | 14,40 | 8,80 | 1,40 | 19,80 | 56,40 | 59,00 | 2,60 | 2,60 | 17,93 |
| | 4,44 | 3,44 | 20,78 | 16,55 | 6,22 | 1,00 | 32,67 | 14,46 | 52,00 | 1,00 | 2,40 | 14,09 |
| 0,6 | 8,00 | 3,90 | 19,70 | 15,00 | 11,40 | 3,00 | 16,40 | 53,40 | 63,50 | 2,40 | 2,00 | 18,06 |
| | 1,00 | 1,00 | 23,60 | 10,10 | 20,20 | 1,00 | 17,20 | 49,10 | 24,00 | 1,40 | 1,00 | 13,60 |
| 0,7 | 8,60 | 6,80 | 21,20 | 16,80 | 8,00 | 2,00 | 19,20 | 60,60 | 66,20 | 2,60 | 1,00 | 19,36 |
| | 1,00 | 1,00 | 27,60 | 51,00 | 13,00 | 1,00 | 21,00 | 43,30 | 35,80 | 1,00 | 1,00 | 17,88 |
| 0,8 | 2,20 | 4,30 | 19,90 | 16,40 | 13,20 | 1,40 | 25,20 | 66,60 | 67,00 | 2,40 | 2,20 | 20,07 |
| | 1,00 | 1,30 | 21,20 | 38,40 | 5,50 | 1,00 | 28,20 | 52,60 | 30,30 | 1,00 | 1,00 | 16,50 |
| 0,9 | 8,00 | 4,40 | 15,10 | 16,80 | 11,40 | 1,40 | 33,80 | 65,40 | 77,70 | 1,60 | 1,20 | 21,53 |
| | 1,00 | 1,10 | 14,00 | 23,70 | 19,60 | 1,00 | 35,40 | 56,20 | 37,00 | 1,00 | 1,00 | 17,36 |
| 1,0 | 7,20 | 4,20 | 14,30 | 19,20 | 17,40 | 2,20 | 30,80 | 76,00 | 66,00 | 2,00 | 2,20 | 21,95 |
| | 1,00 | 1,60 | 27,10 | 21,70 | 29,20 | 1,00 | 30,50 | 45,00 | 40,60 | 1,00 | 1,00 | 18,15 |

C45
ADP
Performance Superior ADP

Tabela 6.4: Tamanho das Árvores Obtidas.

reforçada verificando que para a base Esporte — uma base construída para Árvores de Decisão e que não apresenta fortes relacionamentos entre atributos — o sistema obteve melhores resultados para praticamente todas as situações.

Alguns melhoramentos e extensões podem ser realizados no modelo de ADP. Por exemplo, melhorar o critério de parada do sistema — mecanismo de pré-poda —, pois o critério utilizado atualmente pelo sistema é simples — ver Seção 7.4. É possível que critérios baseados no ganho de informação, ou até mesmo uma estratégia de pós-poda, possam melhorar a performance do sistema nos casos onde são encontradas pequenas taxas de exemplos inconsistentes.

## 6.5 Considerações Finais

Nesta Seção foi apresentado o modelo de inferência paraconsistente em árvores de decisão. Para tanto, definições lógicas dos elementos básicos envolvidos na tarefa de aprendizado foram propostas, obedecendo uma sintaxe semelhante à utilizada pela Lógica Paraconsistente — Capítulo 4.

Verificou-se que o comportamento do método de inferência depende do tipo de questionamento realizado:

1. medir a consistência de determinada classe associada a um exemplo;
2. encontrar a classe mais adequada para o exemplo.

Estas duas formas de inferência podem ser utilizadas para a realização de tarefas distintas. Por exemplo, dado um conjunto de exemplos que possuem suas respectivas classes associadas e representam retinas de imagens de assinaturas [Sabourin, 1997], é possível, utilizando a primeira forma de questionamento, obter graus de inconsistência para cada exemplo, ou retina. Esses graus poderiam ser combinados, segundo algum critério, e um grau de inconsistência global seria atribuído ao conjunto de exemplos e, conseqüentemente, à assinatura correspondente. Obviamente, assinaturas de um mesmo autor deveriam retornar um pequeno grau de inconsistência; por outro lado, assinaturas falsas deveriam retornar um alto grau de inconsistência. Pode-se dizer que este método está relacionado à tarefa de verificação. Como pode ser constatado no Capítulo 7, este modelo de verificação foi utilizado em imagens de assinaturas e teve seus resultados comparados com os resultados do sistema REC-SIG.

É possível utilizar a segunda forma de inferência para a tarefa de classificação. Desta forma, em reconhecimento de assinaturas, cada exemplo de retina necessita ser associado à sua classe. Dado um conjunto de exemplos de retinas que representam uma assinatura, é possível verificar para o modelo de assinatura de cada autor

— representado na forma de ADP —, quantas classificações corretas foram obtidas. Posteriormente, o autor cujo modelo obteve maior número de acertos pode ser identificado como autor da assinatura. Esta segunda forma de inferência foi utilizada na comparação do sistema desenvolvido com o sistema C4.5 — Seção 6.4.

# Capítulo 7

# ADP em Verificação de Assinaturas

## 7.1 Considerações Iniciais

Como introduzido em Seções anteriores, é possível utilizar conceitos de Aprendizado de Máquina, Gerenciamento de Incerteza e Lógica Paraconsistente na construção de um sistema de Aprendizado Paraconsistente, capaz de extrair conhecimento útil mesmo quando informações inconsistentes estão presentes na base de dados. Um sistema com tais características pode, portanto, ser empregado no problema de verificação automática de assinaturas manuscritas.

Neste trabalho, a partir dos padrões de treinamento é gerado, para cada autor, um classificador baseado em árvore de decisão. Pretende-se, através desse classificador, caracterizar o modelo da assinatura do autor. Esse modelo pode ser utilizado para validar assinaturas legítimas do autor e rejeitar possíveis falsificações, ver Seção 7.5. O sistema implementado possui as etapas apresentadas na Figura 7.1.

## 7.2 Representação da Base de Assinaturas

Em [Sabourin, 1997] foi descrita uma nova técnica para tratamento de imagens de assinaturas manuscritas baseada em Granulometria Local — sistema RECSIG — Seção 5. Nesse sistema, imagens de assinaturas são subdivididas em pequenas regiões — retinas — sobre as quais são aplicadas sucessivas operações morfológicas. Através deste processo, é possível obter padrões referentes a cada retina, onde cada padrão apresenta apenas três características básicas: média, variância e inclinação. Desta forma, cada assinatura é constituída de um conjunto de padrões referentes às retinas da assinatura. Esta técnica foi aplicada sobre uma base de imagens que possui assinaturas de 20 autores, onde cada autor contribuiu com 40 assinaturas. Portanto, dispõe-se de 800 assinaturas na base.


Figura 7.1: Etapas do Processo de Reconhecimento.

## 7.3 Transformação nos Dados

Os experimentos realizados neste trabalho utilizaram a mesma base de assinaturas apresentada em [Sabourin, 1997]. Além disso, os padrões de retinas foram extraídos utilizando-se o sistema RECSIG. A partir dos padrões, duas etapas de transformação são realizadas: *Discretização* e *Cálculo de Fatores Evidenciais*. Objetiva-se, com a

transformação dos padrões, obter padrões de retinas com o seguinte formato:

$$retina_i(v_1, ..., v_n) : [Crença, Descrença]$$

tal que, $retina_i$ é a classe do padrão, ou exemplo; $v_j$ é o valor para o atributo discretizado $j$; $n$ é o número de características, ou atributos e, *Crença* e *Descrença* são valores numéricos pertencentes ao intervalo [0,1]. As evidências *Crença* e *Descrença* podem ser interpretadas da seguinte forma:

- *Crença*: acredita-se com grau *Crença* que o conjunto de valores $< v_1, ..., v_n >$ representa uma classe $retina_i$;
- *Descrença*: acredita-se com grau *Descrença* que o conjunto de valores $< v_1, ..., v_n >$ não representa uma classe $retina_i$.

A primeira etapa de transformação nos dados, *Discretização* [Pfahringer, 1995], possui fundamental importância para o sistema implementado, pois pode provocar dois efeitos sobre os dados:

- padrões que possuem valores semelhantes para todos os seus atributos tendem a ser transformados em padrões idênticos. Desta forma, padrões que são semelhantes, mas que pertencem à classes distintas tornam-se inconsistentes;
- um pequeno número de valores para cada atributo torna possível a utilização do classificador Naïve Bayes [Mitchell, 1997] para o cálculo de probabilidades empregadas na obtenção de fatores evidenciais, que será visto a seguir.

Após a discretização, a última etapa de transformação a ser executada sobre os padrões é o Cálculo dos *Fatores Evidenciais* — obtenção dos valores de *Crença* e *Descrença* de cada retina. Pode-se utilizar o formalismo introduzido na *Teoria dos*

*Fatores de Certeza* — Seção 3.4 — para o cálculo dos fatores evidenciais $Crença = MC[h, e]$ e $Descrença = MD[h, e]$.

As duas etapas de tranformação serão abordadas detalhadamente nas Seções posteriores.

### 7.3.1 Discretização

Muitos algoritmos de AM e Data Mining foram construídos para manipular bases de dados que possuíssem apenas atributos discretos [Richeldi, 1995] [Kerber, 1992] — Apêndice A.3. Porém, a maioria dos domínios de aplicação apresentam variáveis que possuem infinitos valores numéricos possíveis — atributos contínuos [Catlett, 1991]. Dessa forma, tornou-se necessário transformar esses atributos contínuos em atributos que possuem apenas alguns valores discretos.

Algoritmos de discretização podem ser supervisionados ou não-supervisionados. Algoritmos não-supervisionados são os mais simples, pois agrupam valores de um determinado atributo em intervalos sem levar em consideração a classe dos exemplos. Exemplos de algoritmos que empregam esta técnica são *intervalos do mesmo tamanho*, *intervalos baseados em freqüência* e *k-médio*. Algoritmos com tal característica tendem a apresentar maior perda de informação e, portanto, podem afetar de forma mais acentuada a performance de um sistema classificador [Pfahringer, 1995].

Vários algoritmos que consideram a informação da classe dos exemplos — algoritmos supervisionados — foram descritos na literatura. Em [Pfahringer, 1995] verificou-se que métodos baseados em entropia — Seção 2.2.3 — podem aumentar consideravelmente a precisão do classificador Naïve Bayes em alguns domínios de aplicação, pois tendem a manter valores do atributo em um mesmo intervalo quando as classes dos exemplos são iguais. Vários algoritmos supervisionados para discretização podem ser citados, por exemplo, discretização baseada no C4.5 [Kohavi, 1996], 1R [Dougherty, 1995], D-2 [Catlett, 1991], ChiMerge [Kerber, 1992], etc.

Dentre os benefícios da discretização aplicada como uma etapa de pré-processamento, em relação à tarefa de aprendizado, pode-se citar:

- aumento no grau de generalização do sistema, uma vez que valores contínuos tendem a diminuir o grau de generalização;
- tornar o conhecimento extraído mais compreensível. Por exemplo, árvores de decisão tendem a ser menores e as regras geradas mais inteligíveis;
- a precisão do sistema pode ser aumentada em alguns domínios de aplicação;
- no contexto de árvore de decisão, o tempo de processamento empregado na tarefa de aprendizado tende a ser muito menor, pois como demonstrado em [Catlett, 1991] o tempo gasto no processo de discretização somado ao tempo de aprendizado sobre uma base discretizada pode ser 10 vezes menor do que o tempo utilizado para treinamento sobre uma base não discretizada.

Estratégias de implementação de algoritmos de discretização são empregadas de forma *top-down* ou *bottom-up*. Algoritmos *top-down* iniciam com um único intervalo — ao qual estão associados todos os valores do atributo — e realizam sucessivos particionamentos até que um determinado critério de parada seja satisfeito. Por outro lado, algoritmos *bottom-up* iniciam com $n$ intervalos, onde cada intervalo corresponde a um valor do atributo. Através de operações de união entre os intervalos — merging — que satisfazem uma determinada condição, novos intervalos são gerados, até que um critério de parada, por exemplo, número mínimo de intervalos, seja satisfeito.

**Discretização Através do Particionamento Recursivo Baseado em Mínima Entropia**

Como descrito em [Dougherty, 1995] [Pfahringer, 1995], métodos de discretização baseados em mínima entropia podem aumentar sensivelmente a precisão do classificador Naïve Bayes. Além disso, são de simples implementação e diversos procedimentos implementados no algoritmo C4.5 podem ser reutilizados.

O método é iniciado ordenando o conjunto $S$ de valores para o atributo. Cada valor $T$ de $S$ é considerado um potencial ponto de corte para o intervalo. Portanto, para cada $T_i \in S$ é calculada a entropia de informação obtida pelo particionamento em $T_i$ através da seguinte fórmula:

$$E(A, T_i, S) = \frac{|S_1|}{|S|} \times Entropia(S1) + \frac{|S_2|}{|S|} \times Entropia(S_2) \tag{7.1}$$

tal que, $S_1$ é o subconjunto de $S$ formado por valores menores ou iguais a $T_i$ e $S_2$ é o subconjunto formado por valores maiores que $T_i$, $A$ é o atributo, e *Entropia* é a definida na Seção 2.2.3. O valor $T_{min}$, que minimiza o valor da entropia, é escolhido como ponto de corte. O processo é realizado recursivamente sobre os subconjuntos $S_1$ e $S_2$, até que um critério de parada seja satisfeito. O critério de parada é dado por:

$$ganho(A, T_i, S) < \frac{\log_2(N-1)}{N} + \frac{\nabla(A, T_i, S)}{N} \tag{7.2}$$

tal que, $N$ é o número de exemplos de $S$.

O ganho é definido por:

$$ganho(A, T_i, S) = Entropia(S) - E(A, T_i, S) \tag{7.3}$$

$\nabla$ é calculado através da seguinte fórmula:

$$\nabla(A, Ti, S) = \log_2(3^k - 2) - [k \times Entropia(S) - k_1 \times Entropia(S_1) - k_2 \times Entropia(S_2)] \tag{7.4}$$

tal que, $k_j$ é o número de classes presentes em $S_j$.

Além das características já mencionadas sobre este método de discretização, outras vantagens podem ser encontradas em relação a alguns métodos. Por exemplo, não é necessário fornecer o número de intervalos desejado ou qualquer outra informação sobre o número de intervalos. Além disso, como cada subintervalo é considerado independentemente, regiões que apresentam grande variabilidade tendem a ser particionadas de forma acentuada, enquanto regiões com baixa entropia possuirão menos intervalos [Dougherty, 1995].

### Dados Inconsistentes no Problema de Verificação de Assinaturas

Como introduzido anteriormente, o processo de discretização pode gerar exemplos de treinamento inconsistentes devido à grande redução do espaço de tuplas. A representação dos exemplos utilizada neste trabalho para o problema de verificação de assinaturas, tende a acentuar esse comportamento. Cada conjunto de treinamento utilizado no problema de verificação apresenta um grande número de classes — referente às retinas que possuem ativação. Além disso, há um pequeno número de atributos — 3 — para representar cada retina.

O efeito do processo de discretização sobre os dados referentes ao problema de assinatura pode ser verificado na Tabela 7.1.

As bases geradas a partir de todos os ESs — Elementos Estruturantes — utilizados apresentaram comportamento semelhante quando submetidas ao processo de discretização. Antes da discretização, as bases apresentavam baixas taxas de exemplos inconsistentes — em geral, menos de 10%. Porém, as mesmas bases dis-

| Percentuais de Inconsistência | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Horizon. | | Vertical | | Diag. 45° | | Diag. 135° | | Circular | |
| Autor | S/ | C/ | S/ | C/ | S/ | C/ | S/ | C/ | S/ | C/ |
| A | 15.35 | 99.02 | 10.05 | 98.92 | 9.60 | 99.51 | 9.80 | 99.02 | 5.74 | 99.90 |
| B | 7.47 | 96.91 | 5.46 | 95.81 | 3.67 | 92.07 | 4.34 | 93.17 | 2.11 | 92.51 |
| C | 8.25 | 96.79 | 5.61 | 99.82 | 6.14 | 97.68 | 6.05 | 98.75 | 3.24 | 100.0 |
| D | 11.55 | 98.43 | 6.30 | 98.6 | 6.52 | 99.53 | 6.60 | 99.38 | 2.38 | 100.0 |
| E | 7.38 | 98.83 | 5.86 | 100.0 | 6.20 | 100.0 | 5.44 | 99.76 | 3.45 | 100.0 |
| F | 13.25 | 90.62 | 6.86 | 85.85 | 6.13 | 94.24 | 7.22 | 88.97 | 3.49 | 93.04 |
| G | 10.15 | 96.70 | 5.54 | 100.0 | 6.40 | 98.11 | 6.25 | 99.53 | 2.42 | 99.84 |
| H | 7.83 | 99.46 | 6.42 | 99.89 | 6.80 | 100.0 | 5.82 | 99.89 | 3.81 | 99.89 |
| I | 8.82 | 99.57 | 7.35 | 99.46 | 5.89 | 99.03 | 6.49 | 100.0 | 3.08 | 99.89 |
| J | 8.23 | 91.60 | 5.22 | 92.91 | 5.22 | 92.12 | 4.18 | 97.64 | 2.22 | 91.07 |
| K | 9.59 | 90.88 | 6.65 | 91.18 | 4.33 | 93.00 | 4.64 | 93.61 | 1.85 | 93.92 |
| L | 18.85 | 94.83 | 13.55 | 97.87 | 11.78 | 95.74 | 12.96 | 93.31 | 5.59 | 99.69 |
| M | 11.31 | 98.70 | 8.14 | 96.91 | 5.61 | 96.91 | 5.93 | 98.53 | 2.84 | 97.88 |
| N | 7.71 | 96.40 | 5.11 | 99.20 | 4.20 | 97.40 | 4.01 | 98.80 | 2.10 | 99.80 |
| O | 7.53 | 99.77 | 4.93 | 99.44 | 6.02 | 99.77 | 4.63 | 99.66 | 3.24 | 99.77 |
| P | 5.39 | 100.0 | 4.63 | 98.86 | 4.16 | 99.81 | 3.60 | 99.62 | 1.98 | 100.0 |
| Q | 10.56 | 98.08 | 7.24 | 98.25 | 7.07 | 99.65 | 7.24 | 97.20 | 3.32 | 99.82 |
| R | 10.51 | 96.57 | 4.18 | 97.71 | 4.55 | 97.52 | 5.76 | 96.19 | 1.58 | 98.28 |
| S | 12.44 | 98.95 | 6.99 | 98.95 | 8.44 | 98.95 | 7.99 | 99.12 | 4.45 | 99.82 |
| T | 7.32 | 99.69 | 5.94 | 99.91 | 5.90 | 99.91 | 5.72 | 99.90 | 3.39 | 99.91 |

O símbolo S/ equivale a "Sem Discretização"
O símbolo C/ equivale a "Com Discretização"

Tabela 7.1: Percentuais de Inconsistência Encontrados nas Bases de Dados de Assinaturas Geradas de Acordo com Cada Elemento Estruturante.

cretizadas apresentaram taxas de inconsistência próximas a 100%[1]. A Figura 7.2 apresenta a relação do percentual de inconsistência, entre bases discretizadas e não discretizadas, produzidas a partir do ES horizontal.

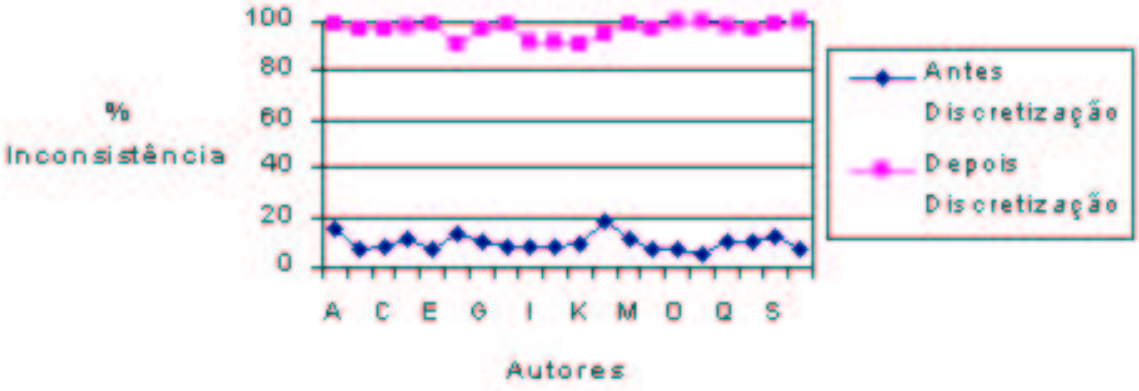



Figura 7.2: Percentual de Inconsistência de Bases Discretizadas e não Discretizadas.

As elevadas taxas de inconsistência obtidas e o grande número de classes tornam o problema de classificação muito difícil. Neste trabalho, pretende-se utilizar o sistema ADP para calcular o fator de inconsistência de cada um dos exemplos que representam as retinas de uma assinatura. Dessa forma, é possível combinar tais fatores de modo a obter-se um grau geral de inconsistência para uma determinada assinatura. O grau geral de inconsistência obtido é utilizado para validar ou rejeitar a assinatura, ou seja, classificá-la como legítima ou falsa. O modelo completo de verificação de assinaturas será abordado em detalhes em Seções posteriores.

### 7.3.2 Obtenção dos Fatores Evidenciais

Através do processo de discretização descrito anteriormente, são obtidos exemplos no seguinte formato:

[1]É importante lembrar que neste trabalho considera-se inconsistente uma base que apresenta exemplos cujos valores dos atributos previsores são idênticos mas diferem quanto ao valor do atributo meta. Portanto, uma base $B$ é considerada 100% inconsistente quando para qualquer exemplo $e \in B$ existe um exemplo $e\prime \in B$ tal que $e\prime$ é inconsistente em relação a $e$.

$$retina_i(v_1, ..., v_n)$$

tal que, cada $v_j$ representa um valor discreto para o atributo $j$. Pode-se considerar, para o formato anterior, a seguinte interpretação:

$$\textbf{se} < v_1 \wedge ... \wedge v_n > \textbf{então } retina_i$$

tal que, $e =< v_1 \wedge ... \wedge v_n >$ é uma seqüência de evidências conjuntivas e $retina_i$ é a hipótese. Portanto, pode-se utilizar o formalismo introduzido na Seção 3.4 — página 37, Equações 3.23 e 3.24 — para o cálculo de $Crença = MC[h, e]$ e $Descrença = MD[h, e]$, com o objetivo de transformar o exemplo para o seguinte formato:

$$retina_i(v_1, ..., v_n) : [Crença, Descrença]$$

É importante ressaltar, que a probabilidade condicional $p(h|e)$ utilizada no cálculo de Crença e Descrença pode ser obtida através da regra de Bayes — Seção 3.2, página 28, Equação 3.6 — e a probabilidade condicional $p(e|h)$, a partir do Classificador Naïve Bayes — Seção 2.5.

A seguir é apresentado um exemplo de como são realizadas as computações de $Crença$ — MC — e $Descrença$ — MD — para exemplos de treinamento já discretizados.

**Exemplo 7.3.1** A Tabela 7.2 apresenta uma pequena base $E$ de retinas agrupadas de acordo com 7 assinaturas. São consideradas hipoteticamente, assinaturas com poucas retinas (3 ou 4) para tornar o exemplo compreensível. As assinaturas podem apresentar número variável de retinas, uma vez que na aplicação real, cada assinatura pode cobrir, ou ativar, retinas diferentes, pois apenas as retinas ativadas — que possuem pixels ativos — são consideradas. Algumas informações adicionais sobre o conjunto de treinamento apresentado são colocadas a seguir:

$$
\begin{aligned}
d_1 &= \{\text{cinco, seis, sete, oito}\} \\
d_2 &= \{\text{um, dois, tres, quatro}\} \\
d_3 &= \{\text{um, dois, tres}\} \\
|E| &= 23
\end{aligned}
$$

tal que, $d_i$ é o domínio do atributo $a_i$ e $|E|$ é o número de exemplos de treinamento do conjunto $E$.

A partir do conjunto de treinamento $E$ — Tabela 7.2 —, é possível computar as probabilidades apresentadas na Tabela 7.3. O cálculo da probabilidade condicional através do classificador Naïve Bayes para o primeiro exemplo de treinamento é apresentado na Tabela 7.4. Utilizando as probabilidades da Tabela 7.4 é possível obter a probabilidade condicional através da Regra de Bayes da seguinte forma:

$$P(retina_{18}|e_1) = \frac{0.1739 \times \ 0.25}{0.1739 \times 0.25 + 0.1739 \times 0.072 + 0.3043 \times 0.0029} = 0.7243 \tag{7.5}$$

O valor obtido pela Equação 7.5 pode ser utilizado no cálculo de $MC$ e $MD$:

$$
\begin{aligned}
MC_{e_1} &= \frac{0.7243 - 0.1739}{1 - 0.1739} = 0.6662 \\
MD_{e_1} &= 0.0
\end{aligned}
$$

uma vez que $P(retina_{18}|e_1) = 0.7243 > P(retina_{18}) = 0.1739$.

## 7.4 Geração dos Classificadores

Dado um conjunto de imagens de assinaturas de um mesmo autor, é possível, após a utilização do sistema RECSIG [Sabourin, 1997] e a aplicação das transformações descritas anteriormente, gerar bases de dados com o mesmo formato do Exemplo

| $e$ | Exemplo |
|---|---|
| $e_1$ | $retina_{18}$(oito,um,dois) |
| $e_2$ | $retina_{22}$(sete,dois,tres) |
| $e_3$ | $retina_{28}$(sete,dois,um) |
| $e_4$ | $retina_{19}$(sete,quatro,tres) |
| $e_5$ | $retina_{22}$(sete,dois,um) |
| $e_6$ | $retina_{28}$(oito,um,um) |
| $e_7$ | $retina_{34}$(cinco,quatro,dois) |
| $e_8$ | $retina_{22}$(oito,um,tres) |
| $e_9$ | $retina_{28}$(sete,dois,um) |
| $e_{10}$ | $retina_{34}$(cinco,quatro,tres) |
| $e_{11}$ | $retina_{18}$(oito,um,um) |
| $e_{12}$ | $retina_{19}$(cinco,quatro,tres) |
| $e_{13}$ | $retina_{28}$(sete,dois,tres) |
| $e_{14}$ | $retina_{19}$(oito,um,um) |
| $e_{15}$ | $retina_{22}$(oito,um,dois) |
| $e_{16}$ | $retina_{28}$(sete,dois,um) |
| $e_{17}$ | $retina_{18}$(oito,um,um) |
| $e_{18}$ | $retina_{19}$(cinco,quatro,tres) |
| $e_{19}$ | $retina_{22}$(oito,um,um) |
| $e_{20}$ | $retina_{28}$(sete,dois,um) |
| $e_{21}$ | $retina_{18}$(oito,um,um) |
| $e_{22}$ | $retina_{28}$(sete,dois,dois) |
| $e_{23}$ | $retina_{34}$(seis,tres,dois) |

Tabela 7.2: Conjunto de Treinamento $E$.

| Probabilidades | $retina_{18}$ | $retina_{19}$ | $retina_{22}$ | $retina_{28}$ | $retina_{34}$ |
|---|---|---|---|---|---|
| $P(a_1 = cinco\|classe)$ | 0.0 | 0.5 | 0.0 | 0.0 | 0.67 |
| $P(a_1 = seis\|classe)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.33 |
| $P(a_1 = sete\|classe)$ | 0.0 | 0.25 | 0.4 | 0.857 | 0.0 |
| $P(a_1 = oito\|classe)$ | 1.0 | 0.25 | 0.6 | 0.143 | 0.0 |
| $P(a_2 = um\|classe)$ | 1.0 | 0.25 | 0.6 | 0.143 | 0.0 |
| $P(a_2 = dois\|classe)$ | 0.0 | 0.0 | 0.4 | 0.857 | 0.0 |
| $P(a_2 = tres\|classe)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.33 |
| $P(a_2 = quatro\|classe)$ | 0.0 | 0.75 | 0.0 | 0.0 | 0.67 |
| $P(a_3 = um\|classe)$ | 0.75 | 0.25 | 0.4 | 0.714 | 0.0 |
| $P(a_3 = dois\|classe)$ | 0.25 | 0.0 | 0.2 | 0.143 | 0.67 |
| $P(a_3 = tres\|classe)$ | 0.0 | 0.75 | 0.4 | 0.143 | 0.33 |
| $P(classe)$ | 0.1739 | 0.1739 | 0.2174 | 0.3043 | 0.1304 |

Tabela 7.3: Probabilidades Calculadas a Partir de $E$.

| Classe | $P(e_1\|classe)$ "Naïve Bayes" |
|---|---|
| $retina_{18}$ | $1.0 \times 1.0 \times 0.25 = 0.25$ |
| $retina_{19}$ | $0.25 \times 0.25 \times 0.0 = 0.0$ |
| $retina_{22}$ | $0.6 \times 0.6 \times 0.2 = 0.072$ |
| $retina_{28}$ | $0.143 \times 0.143 \times 0.143 = 0.0029$ |
| $retina_{34}$ | $0.0 \times 0.0 \times 0.67 = 0.0$ |

Tabela 7.4: Cálculo da Probabilidade Condicional Através do Classificador Naïve Bayes.

6.2.2. Pode-se, portanto, gerar um classificador baseado em Árvore de Decisão Paraconsistente — Seções 2.2 e 6.2.2 — que represente o modelo para tal base de dados.

Como pôde-se perceber no Exemplo 6.2.2, nós-folha podem apresentar exemplos de classes diferentes. Portanto, torna-se necessária, uma modificação em relação ao algoritmo original de AD. Foi implementado no sistema um novo critério de parada que representa um mecanismo de pré-poda — Seção 2.4 —, descrito a seguir.

Dado um parâmetro $F$ fornecido como entrada para o sistema, o novo critério de parada é:

> Se $E$ é um conjunto não-vazio de exemplos e há um subconjunto $E_i \subseteq E$ de exemplos pertencentes à uma mesma classe, tal que $\frac{|E_i|}{|E|} \geq F$, então forme uma folha e encerre, senão continue a geração da árvore.

Como pode-se perceber, $F$ pertence ao intervalo $[0, 1]$ e refina o grau de generalização da árvore. A árvore gerada tende a ser mais específica, portanto, mais complexa, à medida em que o valor de $F$ aumenta. Por outro lado, a árvore tende a ser mais genérica e simples, à medida em que o valor de $F$ diminui.

Esse comportamento pode ser observado através das Figuras 7.3 e 7.4. As árvores ilustradas foram produzidas a partir de uma mesma base de dados, alterando-se, apenas, o fator de poda. As árvores das Figuras 7.3 e 7.4 foram geradas com fatores de poda de 0.8 e 0.85, respectivamente. A base utilizada para exemplificação chama-se IRIS, uma base muito conhecida na comunidade de Aprendizado de Máquina. Tal base apresenta 150 exemplos formados por 4 atributos de valores numéricos inteiros.

Intuitivamente, quando $F$ é igual a 1.0, o comportamento do sistema é o mesmo em relação ao algoritmo básico de AD. A estratégia de poda adotada teve duas motivações principais:

- a primeira motivação, deve-se à presença de padrões inconsistentes. Torna-se

```
| ?- idt(0.8,'/fabricio/assinatura/bases/iris/iris.tre').

yes
| ?- show_decision_tree.

a4=quatro ==> class=tres:[1.0,0.0] : 0.0
          ==> class=dois:[0.0,0.8852] : 0.1148

a4=tres ==> class=dois:[1.0,0.0] : 0.0
        ==> class=tres:[1.0,0.9937] : 0.9937

a4=dois ==> class=dois:[1.0,0.0] : 0.0
a4=um ==> class=um:[1.0,0.0] : 0.0
a4=cinco ==> class=tres:[1.0,0.0] : 0.0
```

Figura 7.3: Árvore Simples.

necessário a criação de folhas que não apresentam apenas exemplos de uma mesma classe — como ilustrado nas Figuras 7.3 e 7.4;

- a segunda motivação, em relação à escolha de pré-poda, deve-se à maior eficiência obtida com o algoritmo em relação à pós-poda [Quinlan, 1993].

Nos experimentos realizados, não foram feitos testes comparativos em relação à outros critérios de poda, e, portanto, não há como garantir que as árvores obtidas sejam as melhores. Tal estudo está fora do escopo deste trabalho e poderia ser considerado, futuramente, como tema de novas pesquisas. Outra consideração importante está relacionada ao critério de escolha de atributo. O critério implementado no sistema é o mesmo apresentado na Seção 2.2.3 — *ganho_médio* de informação.

Devido à estratégia de poda adotada, em alguns domínios pode ser necessário um conhecimento prévio dos dados para a atribuição adequada do valor de $F$. Porém, para verificação de assinaturas esse problema não ocorre. Isso porque devido ao grande número de classes, à redução acentuada do espaço de tuplas — provocada pela discretização — e ao pequeno número de atributos, as folhas tendem a apresentar exemplos de várias classes. Dessa forma, dificilmente uma classe representará a grande maioria dos exemplos em uma folha.

```
| ?- idt(0.85,'/fabricio/assinatura/bases/iris/iris.tre').

yes
| ?- show_decision_tree.

a4=quatro and ==> class=tres:[1.0,0.0] : 0.0
              ==> class=dois:[0.0,0.8852] : 0.1148

a4=tres and a3=cinco ==> class=tres:[1.0,0.0] : 0.0
        a3=dois ==> class=dois:[1.0,0.0] : 0.0
        a3=quatro and a2=um ==> class=tres:[0.0,0.8978] : 0.1022
                  a2=tres ==> class=dois:[0.0753,0.0] : 0.9247
                  a2=quatro ==> class=dois:[0.0288,0.0] : 0.9712
                  a2=dois and a1=cinco and ==> class=dois:[0.3077,0.0] : 0.6923
                                           ==> class=tres:[0.1923,0.0] : 0.8077

                              a1=quatro ==> class=dois:[0.6962,0.0] : 0.3038

        a3=tres and ==> class=dois:[0.9955,0.0] : 0.0045
                    ==> class=tres:[0.0,0.9937] : 0.0063

a4=dois ==> class=dois:[1.0,0.0] : 0.0
a4=um ==> class=um:[1.0,0.0] : 0.0
a4=cinco ==> class=tres:[1.0,0.0] : 0.0
```

Figura 7.4: Árvore Complexa.

## 7.5 Verificação de Assinaturas

Como mencionado anteriormente, a base de assinaturas $B$ é constituída por 20 autores, onde cada autor contribuiu com 40 assinaturas, portanto, $|B| = 20 \times 40 = 800$. Com o objetivo de permitir a comparação dos resultados obtidos com os apresentados em [Sabourin, 1997], a base $B$ foi dividida em assinaturas de treinamento $R$ e teste $T$, tal que $|R| = |T| = 400$. Ou seja, para cada autor, foram utilizadas 20 assinaturas para treinamento e 20 para teste, ver Seção 7.7. Para simplificação, $R_i$ e $T_i$ referem-se ao conjunto de assinaturas de treinamento e teste do autor $i$, respectivamente. A partir de $R_i$ é gerado um classificador $C_i$. O classificador obtido é capaz de atribuir evidências às retinas de uma assinatura, conforme mostra a Figura 7.5.

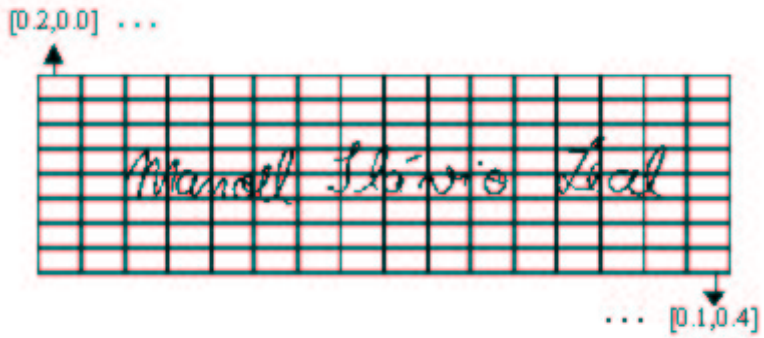



Figura 7.5: Assinatura com Fatores Evidenciais de Retinas Calculados.

### 7.5.1 Obtenção do Grau de Inconsistência de uma Assinatura

A partir das evidências das retinas de uma assinatura, é possível combiná-las de forma a obter-se um grau geral de inconsistência para a assinatura. A função de combinação deve atribuir um grau de inconsistência maior para assinaturas que possuam retinas com graus de crença [0.0,0.0], uma vez que tais retinas não foram reconhecidas pelo classificador.

Intuitivamente, poder-se-ia calcular a inconsistência de uma assinatura utilizando a operação $sup\{[C_1,D_1],...,[C_n,D_n]\}$, tal que $C_i$ e $D_i$ são os valores de crença e descrença atribuídos à retina $i$, respectivamente. Porém, devido à restrição que a ocorrência da evidncia [0.0,0.0] deve aumentar a inconsistência de uma assinatura, a operação *sup* não pode ser aplicada diretamente. Por exemplo, dada uma assinatura $A_n$, tal que $n$ é o número de retinas, quando $\frac{n}{2}$ retinas de $A$ possuem evidências [0.0,0.0], o grau mínimo de inconsistência da assinatura é 0.5. Isto ocorre porque metade da imagem não pôde ser reconhecida pelo classificador. Pode-se considerar, também, que tal assinatura é totalmente inconsistente, e, portanto, deve ser rejeitada. Partindo desse princípio, foi desenvolvida a seguinte equação de combinação das evidências das retinas de uma determinada assinatura:

$$G(A) = I/S(sup\{[C_1, D_1], ..., [C_N, D_N]\}) \times \left(1 - \frac{(2 \times N_{[0.0,0.0]})}{N}\right) + \frac{2 \times N_{[0.0,0.0]}}{N} \tag{7.6}$$

tal que, $N_{[0.0,0.0]}$ é o número de retinas que possuem crenças [0.0,0.0], $N$ é o número de retinas da imagem e $I/S$ e $sup$ são operadores que representam grau de inconsistência/subdeterminação e supremo, respectivamente — Seção 4.2.

A Equação 7.6 garante que uma assinatura que apresenta metade das retinas com grau de crença [0.0,0.0] seja rejeitada. Por exemplo, dada uma assinatura $A$ que apresenta $N = 128$ e 64 retinas com grau de crença [0.0,0.0], a substituição dos valores na Equação 7.6 é dada a seguir:

$$G(A) = I/S(sup\{...\}) \times (1 - 1) + \frac{128}{128}$$

Verifica-se que a assinatura apresenta inconsistência total, ou seja, 1.0. Da mesma forma, quando o percentual de retinas que apresentam evidências [0.0,0.0] é maior do que 0.5 — 50% — o sistema também atribui fator máximo de inconsistência para a assinatura. A relação entre o grau de inconsistência de uma assinatura e a percentagem de retinas que possuem evidências [0.0,0.0] pode ser visualizada na Figura 7.6.

### 7.5.2 Obtenção do Limiar de Rejeição

Na Seção anterior pôde-se verificar que assinaturas apresentando um alto grau de inconsistência necessitam ser rejeitadas. Obviamente, nem sempre assinaturas falsas apresentarão valor máximo de inconsistência. Portanto, é necessário estabelecer um limiar $L$ que possa validar assinaturas com $G \leq L$ e rejeitar assinaturas com $G > L$. O limiar de inconsistência individual para cada autor é calculado automaticamente, utilizando-se os exemplos de treinamento, como ilustrado na Figura 7.7.

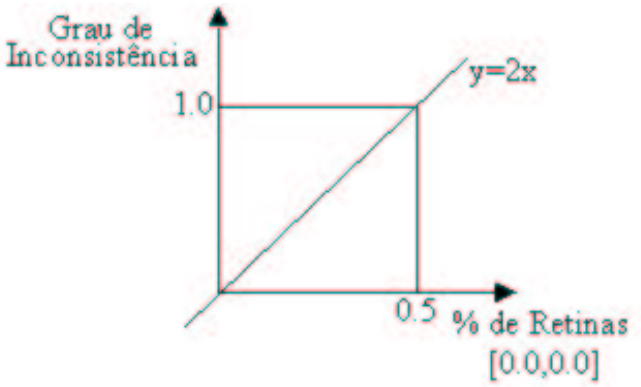



Figura 7.6: Relação entre o Grau de Inconsistência de uma Assinatura e o Percentual de Retinas [0.0,0.0].

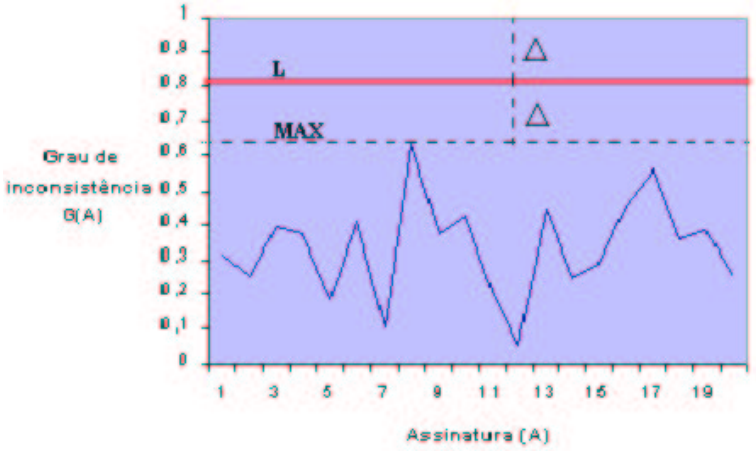



Figura 7.7: Definição do Limiar de Inconsistência para Assinaturas de Treinamento de um Determinado Autor.

Intuitivamente, poder-se-ia utilizar o maior grau de inconsistência obtido nas assinaturas de treinamento como o limiar de inconsistência. Porém, na maioria das aplicações que envolvem o problema de classificação, é comum que assinaturas de teste não sejam perfeitamente reconhecidas pelo modelo obtido a partir do algoritmo de aprendizado. Portanto, é utilizado um valor $\Delta$ para a obtenção do limiar, definido da seguinte forma:

$$\Delta = (1 - MAX)/2 \tag{7.7}$$

tal que, $MAX$ é o maior valor de inconsistência obtido sobre as assinaturas de treinamento.

Logo, a definição de $L$ é dada por:

$$L = \Delta + MAX \tag{7.8}$$

## 7.6 Considerações Sobre a Performance do Sistema

A partir da representação de assinaturas descrita na Seção 7.2 o sistema foi avaliado para cada autor, considerando dois tipos de erro:

- $E_1$: representa a taxa de rejeição de assinaturas legítimas;
- $E_2$: representa a taxa de validação de assinaturas falsas.

O erro médio, denotado $E_t$ é calculado a partir de $(E_1 + E_2)/2$. É importante salientar que tanto a notação quanto a metodologia de teste empregadas estão baseadas nos experimentos relatados em [Sabourin, 1994]. Esta restrição é assumida com o objetivo de tornar possível a comparação dos resultados ilustrados nestes trabalhos.

Como demonstrado em [Sabourin, 1997], os melhores resultados foram obtidos com retinas que apresentavam 50% de sobreposição horizontal em relação às retinas vizinhas. Além disso, o pseudopecstrum aplicado com elementos estruturantes {|,—,/,\,•} apresentou uma significativa diminuição no erro médio.

| ES | $E_1$ | $E_2$ | $E_t$ |
|---|---|---|---|
| — | 0.00 | 0.04 | 0.02 |
| \| | 0.00 | 0.04 | 0.02 |
| / | 0.00 | 0.04 | 0.02 |
| \ | 0.00 | 0.03 | 0.02 |
| • | 0.20 | 0.15 | 0.18 |

(a)

| ES | $E_1$ | $E_2$ | $E_t$ |
|---|---|---|---|
| — | 1.55 | 0.15 | 0.85 |
| \| | 2.67 | 0.48 | 1.58 |
| / | 3.01 | 0.32 | 1.65 |
| \ | 2.07 | 0.31 | 1.19 |

(b)

| ES | $E_1$ | $E_2$ | $E_t$ |
|---|---|---|---|
| — | 2.30 | 2.17 | 2.23 |
| \| | 1.78 | 2.39 | 2.08 |
| / | 2.39 | 3.65 | 3.02 |
| \ | 2.08 | 1.78 | 1.93 |
| • | 1.73 | 3.73 | 2.74 |

(c)

Tabela 7.5: Resultados, em (%), obtidos a partir do Pseudopecstrum Aumentado: (a) NN (25 Iterações), (b) Mínima Distância (25 Iterações) e (c) ADP.

## 7.7 Metodologia de Teste

Como mencionado anteriormente, para cada $R_i$ $(0 \leq i \leq 20)$ foi construído um classificador $C_i$. As assinaturas de teste $T_i$ foram submetidas ao respectivo classificador $C_i$ para que o sistema fosse avaliado em termos de $E_1$. A performance do sistema, em termos de $E_2$, foi avaliada considerando as assinaturas dos 19 autores restantes. Para cada autor restante, 5 assinaturas foram escolhidas randomicamente e submetidas ao classificador $C_i$. Dessa forma, sobre cada $C_i$ foram aplicadas $20 + 95 = 115$ assinaturas. Na Figura 7.8 é apresentado o algoritmo utilizado para avaliação do sistema. O símbolo $base_i$ é utilizado para representar o conjunto de assinaturas do autor $i$.

### 7.7.1 Resultados Obtidos

A seguir são apresentados os resultados obtidos pelo sistema e alguns resultados apresentados em [Sabourin, 1997], ver Tabela 7.5.

A descrição detalhada dos classificadores Vizinho mais Próximo — NN — e Mínima Distância — MMD — pode ser obtida a partir de [Sabourin, 1993]. Pode-se

```
for i=1 to 20 do
  dividir base_i em R_i e T_i
  discretizar R_i
  construir o classificador C_i a partir de R_i
  for j=1 to 20 do
    classificar a assinatura T_ij em C_i
    if assinatura T_ij não é aceita then
      incrementar E_1
    end if
  end for
  Temp={}
  for j=1 to 20 do
    if j¡¿i then
      escolher randomicamente 5 assinaturas de base_j
      armazená-las em Temp
    end if
  end for
  for j=1 to 95 do
    classificar a assinatura Temp_j em C_i
    if Temp_j é aceita then
      incrementar E_2
    end if
  end for
end for
E_t=(E_1+E_2)/2
```

Figura 7.8: Algoritmo de Avaliação e Teste.

verificar que os resultados obtidos a partir da Árvore de Decisão Paraconsistente — ADP —, implementada neste trabalho, são semelhantes aos resultados obtidos pelos modelos de aprendizado baseado em instância — NN e MMD —, implementados em [Sabourin, 1997]. Devido à simplicidade de representação e às deficiências inerentes ao próprio algoritmo de AD, pode-se afirmar que os resultados obtidos são muito interessantes.

Apesar de apresentar performance inferior, em relação aos resultados dos outros classificadores apresentados, o método de ADP elimina sérias deficiências encontradas em tais classificadores. Por exemplo, a classificação de um exemplo nos dois modelos — NN e MMD — requer que todos os exemplos de treinamento, ou seja, todas as assinaturas, estejam disponíveis em memória para que o sistema possa validar uma determinada assinatura. Essa restrição torna impraticável a aplicação de tais técnicas, em problemas reais de verificação e reconhecimento de assinaturas manuscritas. Utilizando ADP essa restrição não existe, pois basta que o classificador do autor — Árvore de Decisão Paraconsistente gerada — esteja carregado para que o sistema possa validar ou não uma assinatura. Uma outra deficiência dos modelos NN e MMD é o tempo necessário para verificação. Mesmo que todas as assinaturas estejam disponíveis em memória, o tempo de processamento necessário para que o sistema possa realizar todos os cálculos de distância é demasiadamente grande. Porém, em ADP, como pode ser verificado na Seção 2.2.1, a classificação de um exemplo é trivial e rápida.

### 7.7.2 Comparação da Performance do Sistema Implementado em Relação aos Elementos Estruturantes Utilizados

A Figura 7.9 apresenta a performance obtida pelo sistema de acordo com todos os ESs — elementos estruturantes — utilizados. Pode-se perceber, claramente, que há

um "trade-off " entre o número de assinaturas de referência e a taxa de $E_t$, uma vez que, quanto menor o número de assinaturas de referência, maior é a taxa de $E_t$ obtida. Esse comportamento é comum em sistemas de Aprendizado de Máquina, pois quando não se dispõe de uma amostra estatisticamente representativa, dificilmente um sistema poderá estimar a função desejada. Em verificação de assinaturas, esse problema é ainda mais acentuado devido à necessidade de se utilizar poucas assinaturas de referência de um determinado autor.

O comportamento do sistema implementado foi muito semelhante para todos os ESs utilizados. Pode-se destacar que a curva do ES Circular apresenta uma taxa de $E_t$ próxima a 2 % com 15 assinaturas de referência. Porém, o ES Diagonal 135° obteve a melhor taxa de $E_t$ utilizando todas as 20 assinaturas de referência.

Em relação aos resultados obtidos em [Sabourin, 1997] — Tabelas 7.5 (a) e 7.5 (b) — pode-se explicar a maior taxa de $E_t$ obtida pelo sistema implementado, com o alto grau de especificidade de algoritmos de Aprendizado de Máquina Baseado em Instância. Além disso, a transformação nos dados — discretização — e a simplicidade da forma de representação do conhecimento descoberto em ADs, tendem a diminuir o grau de especificidade do sistema de aprendizado implementado.

## 7.8 Considerações Finais

Nesta Seção foram descritas todas as etapas envolvidas na construção de um sistema paraconsistente de verificação de assinaturas manuscritas baseado em árvore de decisão. Conceitos de Aprendizado de Máquina, Gerenciamento de Incerteza e Lógica Paraconsistente foram empregados com o objetivo de definir um formalismo capaz de rejeitar assinaturas falsas e validar assinaturas legítimas de um determinado autor. As falsificações foram consideradas como assinaturas de outros autores, e, portanto, representam falsificações "grosseiras". Foi utilizada para os experimentos,

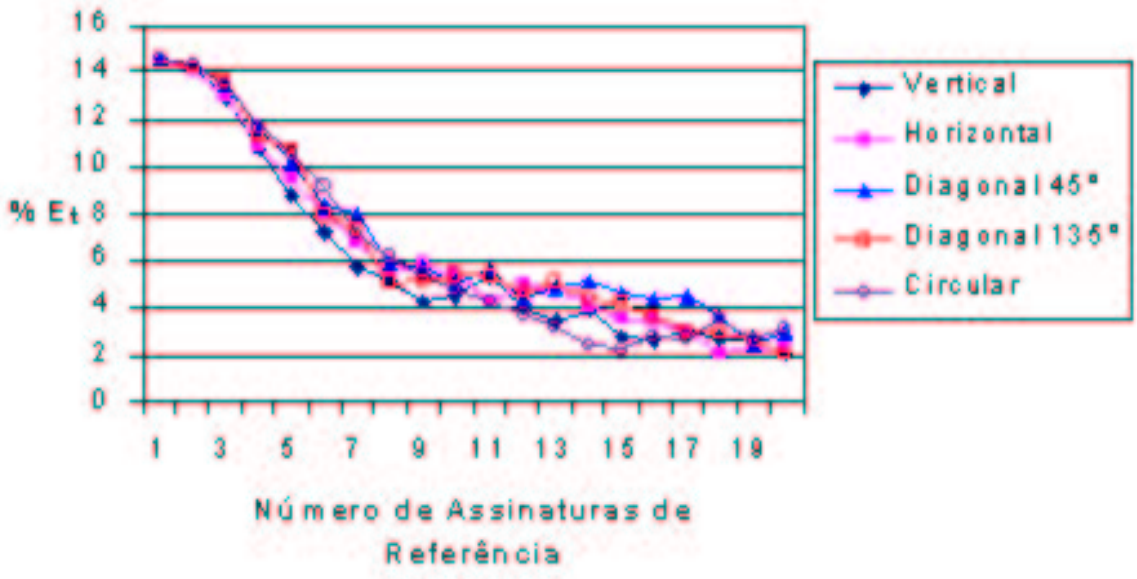



Figura 7.9: Performance do Sistema em Relação aos ESs Utilizados.

a base de 800 imagens de assinaturas de [Sabourin, 1997]. Além disso, um software para tratamento de imagens também foi utilizado — RECSIG (Capítulo 5) — com o objetivo de transformar cada imagem em um conjunto de padrões que representam a descrição de subregiões da imagem.

Constatou-se que, devido às características da aplicação, alterações deveriam ser realizadas no algoritmo básico de AD, tanto no que se refere à geração da árvore, quanto ao procedimento de classificação de um determinado exemplo.

Os resultados obtidos, a partir do classificador proposto, apresentam uma boa taxa de precisão. As vantagens introduzidas com a ADP tornam o modelo de verificação de assinaturas proposto robusto e perfeitamente aplicável à grandes bases de assinaturas. Além disso, trata-se de um modelo genérico, e pode ser aplicado sobre quaisquer bases de dados.

Alguns melhoramentos e extensões ao modelo de ADP podem ser realizados. Por exemplo, o conhecimento compreensível extraído pela árvore pode ser utilizado no sentido de auxiliar o usuário em determinadas tarefas, como por exemplo, na escolha

de outras características para representação das retinas.

# Capítulo 8

# Conclusões

Conceitos de Aprendizado de Máquina e Programação Lógica Evidencial Paraconsistente — PrLE — podem ser utilizados conjuntamente no desenvolvimento de algoritmos capazes de extrair informações úteis e confiáveis sobre dados imperfeitos. Tais algoritmos são de extrema utilidade, uma vez que bases de dados do mundo real estão repletas de impurezas, como por exemplo *inconsistência*, que prejudicam a tarefa de aprendizado.

Em Inteligência Artificial técnicas de Gerenciamento de Incerteza são empregadas na construção de modelos de inferência sobre informações imperfeitas, como por exemplo, Teoria da Probabilidade, Teoria de Dempster-Shafer, Fatores de Certeza e Lógica Fuzzy. Porém, nenhum desses formalismos foi desenvolvido especificamente para manipulação de informações inconsistentes.

Neste trabalho foram empregados conceitos de PrLE para manipulação adequada de informações inconsistentes. A PrLE foi desenvolvida para fornecer interpretações em situações onde ocorram inconsistências. Outros tipos de imperfeições também podem ser manipulados pela LEP, como por exemplo, situações de incerteza ou falta de informações.

Os conceitos de LEP serviram como base para o desenvolvimento de um algoritmo de Aprendizado de Máquina baseado em Árvore de Decisão, capaz de tratar a presença de exemplos inconsistentes em bases de dados. O sistema implementado pode "aprender" um modelo a partir de bases de dados. O modelo obtido — representado através de uma Árvore de Decisão Paraconsistente (ADP) — pode ser utilizado para realizar duas tarefas:

1. *classificação*: atribuir uma classe para um exemplo desconhecido;
2. *cálculo de inconsistência*: atribuir, à classe conhecida de um exemplo, um determinado grau de inconsistência em relação ao modelo obtido.

Verificou-se que a segunda funcionalidade do sistema poderia ser empregada na tarefa de verificação de assinaturas manuscritas. Em [Sabourin, 1997] foi desenvolvido um formalismo capaz de validar ou rejeitar a assinatura manuscrita de um determinado autor através de características locais da assinatura. Ou seja, a imagem da assinatura é dividida em subregiões chamadas retinas. Sobre a imagem de cada retina são aplicadas sucessivas operações morfológicas — Granulometria — e três primitivas são obtidas. Esse processo foi descrito em detalhes no Capítulo 5.

A partir de uma base de imagens de assinaturas de 20 autores, foi gerada, para cada autor, uma base de dados. Um classificador — ADP — também foi construído para cada autor. A idéia se baseia no conceito que para validar, ou rejeitar, a assinatura de um determinado autor não é necessário comparará-la com a assinatura de outros autores.

Dada uma assinatura $A$ de um determinado autor $i$, o classificador $C_i$ é utilizado para calcular fatores evidenciais de *Crença* e *Descrença* para cada retina da assinatura. Posteriormente, é aplicado um operador sobre os fatores evidenciais de todas as retinas e um fator geral de inconsistência é atribuído à assinatura. Esse fator é, posteriormente, comparado com um limiar e a assinatura é validada ou rejeitada.

## 8.1 Resultados Obtidos

O sistema de Aprendizado de Máquina baseado em Árvore de Decisão Paraconsistente desenvolvido foi aplicado no problema de verificação automática de assinaturas. A performance do sistema foi comparada com os resultados apresentados em [Sabourin, 1997].

Foi implementado um procedimento de avaliação com o objetivo de comparar resultados apresentados em [Sabourin, 1997]. Os resultados alcançados pelo sistema foram semelhantes aos alcançados pelos modelos de Aprendizado Baseado em Instância apresentados em [Sabourin, 1997]. Porém, a necessidade de recursos computacionais extremamente poderosos, tornam os algoritmos de classificação utilizados em [Sabourin, 1997] impraticáveis em aplicações reais. A abordagem introduzida com a ADP elimina essa necessidade, e pode ser aplicada a grandes bases de assinaturas.

Com o objetivo de avaliar o sistema, em termos de classificação, foi desenvolvido um procedimento de avaliação ao qual foram submetidos os algoritmos ADP e C4.5. Foram utilizadas nos experimentos diversas bases conhecidas na comunidade de Aprendizado de Máquina. Posteriormente, tais bases foram modificadas de forma a apresentarem exemplos inconsistentes. Sobre cada base de dados, um determinado percentual de exemplos inconsistentes foi gerado. Os exemplos foram escolhidos aleatoriamente e tiveram suas classes alteradas, de forma a tornar a base de dados inconsistente.

Os resultados comparativos entre o C4.5 e a ADP demonstram que para bases de dados que apresentam baixos índices de inconsistência, os algoritmos apresentam resultados semelhantes, com uma pequena superioridade do C4.5. Porém, para a maioria das situações onde grandes quantidades de exemplos inconsistentes estavam presentes, a ADP apresentou melhores resultados.

Dessa forma, pode-se atribuir ao trabalho, pelo menos três contribuições para a comunidade científica. A primeira consiste no desenvolvimento de um algoritmo de Aprendizado de Máquina Paraconsistente. A segunda contribuição consiste em uma nova aplicação de Lógica Paraconsistente — tratamento de exemplos inconsistentes em Aprendizado de Máquina. Finalmente, o sistema desenvolvido representa uma nova abordagem ao problema de verificação automática de assinaturas manuscritas.

## 8.2 Extensões e Trabalhos Futuros

Diversas extensões podem ser realizadas no algoritmo desenvolvido com o objetivo de testar o seu comportamento. Por exemplo, o critério de poda utilizado é simples. Talvez um critério baseado em ganho de informação possa resultar em árvores menores e mais precisas. Outro ponto a ser explorado está relacionado à discretização. Um método menos genérico, desenvolvido espeficamente para o algoritmo de Árvore de Decisão, pode melhorar tanto a eficiência quanto a eficácia do algoritmo.

Novos operadores, utilizados tanto na escolha de uma classe em um determinado nó-folha da árvore, quanto na obtenção do grau geral de inconsistência de uma assinatura, podem ser desenvolvidos. Em ambas as situações, o sistema utiliza a aplicação do operador $MAX$ através do supremo. Outros operadores podem ser testados de forma a obter valores que correspondem a combinações mais uniformes e menos radicais do que $MAX$.

O desenvolvimento de um formalismo capaz de calcular os fatores evidenciais de Crença e Descrença, de acordo com os conceitos da PrLE, também é de extrema utilidade. É importante ressaltar que o formalismo utilizado neste trabalho para o cálculo de evidências — Fatores de Certeza — não pode atribuir à Crença e Descrença valores diferentes de zero ao mesmo tempo, como pode ser verificado nas definições apresentadas na Seção 3.4.

A capacidade de reconhecer dinamicamente a variabilidade dos exemplos — através de *aprendizado incremental* — também é uma extensão interessante a ser realizada no sistema. Essa aplicação é extremamente útil, por exemplo, em reconhecimento e verificação de assinaturas manuscritas, uma vez que assinaturas tendem a sofrer grandes transformações ao longo da vida de uma pessoa.

A utilização da LEP em outros paradigmas de Aprendizado de Máquina, também representa uma importante contribuição para as áreas de pesquisa envolvidas.

# Bibliografia


[Abe, 1992] Abe, J. M., Papavero, N., *Teoria Intuitiva dos Conjuntos*, Makron Books do Brasil Editora Ltda, São Paulo, 1992.

[Anastassopoulos, 1991] Anastassopoulos, V.; Venetsanopoulos, A. N., *The Classification Properties of the Pecstrum and Its Use for Pattern Identification*, Circuits, Systems and Signal Processing, vol. 10, n° 3, pp. 293-326, 1991.

[Ávila, 1996] Ávila, B.C., *Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança*, Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 1996, 133 pg.

[Ávila, 1997] Ávila, B.C., Uma Extensão da Linguagem Prolog para Suportar Programação Lógica Evidencial, Instituto de Estudos Avançados da Universidade de São Paulo, Universidade de São Paulo, São Paulo, 1987.

[Ávila, 1998] Ávila, B.C., *Data Mining*, VI Escola de Informática da SBC Regional Sul, pp. 87-106, 4-8 Maio, 1998.

[Bastos, 1995] Bastos, L. C., *Modelagem Matemática de Assinaturas*, Dissertação de Mestrado, Curso de Pós-Graduação em Enge-

nharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná (CEFET-PR), Curitiba, 1995, 126 pg.

[Benferhat, 1995] Benferhat, S.; Dubois, D.; Prade, H., *How to infer from inconsistent beliefs without revising?*, Proceedings of Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), vol. 2, pp. 1449-1455, Montréal, Québec, Canadá, 20-25 August, 1995.

[Blair, 1987] Blair, H. A.; Subrahmanian, V.S., *Paraconsistent Logic Programming*, LPRG-TR-87-8, Syracuse University, Syracuse, 1987, 24 pg.

[Blair, 1988] Blair, H.A.; Subrahmanian, V.S., Paraconsistent Foundations for Logic Programming, Journal of Non-Classical Logic, 5, 2, pp. 45-73, 1988.

[Bratko, 1990] Bratko, I., Programming for Artificial Intelligence, Addison-Wesley Publishers Ltd., 2 ed., Singapore, 1990.

[Buchanan, 1984] Buchanan, B. G.; Shortliffe, E.H., *Rule-Based Expet Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley Publishing Company, pp. 209-292, USA, 1984.

[Carvalho, 1999] Carvalho, D. R., *Data Mining Através de Indução de Regras e Algoritmos Genéticos*, Dissertação de Mestrado, Mestrado em Informática Aplicada, Pontifícia Universidade Católica do Paraná, Curitiba, 1999, 130 pg.

[Casanova, 1987] Casanova, M.A.; Giorno, F.A.C.; Furtado, A. L. F., *Programação em Lógica e a Linguagem Prolog*, Editora Edgard Blücher LTDA, São Paulo, 1987

[Catlett, 1991] Catlett, J., *On Changing Continuous Attributes Into Ordered Discrete Attributes*, Proc. European Working Session on Learning - EWSL, pp. 164-178, Portugal, March, 1991.

[Cho, 1998] Cho, V.; Wüthrich, B., *Towards Real Time Discovery from Distributed Information Sources*, Pacific-Asia Conf on Knowledge Discovery and Data Mining, Lecture Notes in AI 1394, pp. 376-377, 1998.

[Clark, 1989] Clark, P.; Niblett, T., *The CN2 Induction Algorithm*, Machine Learning Journal, vol. 3, pp. 261-283, Kluwer, Netherlands, 1989.

[Clark, 1991] Clark, P.; Boswell, R., *Rule Induction with CN2: Some Recent Improvements*, Proceeedings of the Fifth European Conference EWSL-91, pp. 151-163, Springer-Verlag, Berlin, 1991.

[Clark, 1993] Clark, P.; Matwin, S., *Using Qualitative Models to Guide Inductive Learning*, Proceedings of 10th International Machine Learning Conference, pp. 49-56, California, 1993.

[Clocksin, 1987] Clocksin, W.F.; Mellish, C. S., *Programming in Prolog*, 3 ed., Springer-Verlag, Germany, 1987.

[da Costa, 1963] da Costa, N.C.A., *Calculs Propositionnels pour les Systèmes Formels Inconsistants*, Compte Rendu Acad. des Sciences (Paris), 257, pp. 3790-3792, 1963.

[da Costa, 1974] da Costa, N.C.A., *On the Theory of Inconsistent Formal Systems*, Notre Dame J. of Formal Logic 15 (1974), pp. 497-510.

[da Costa, 1991] da Costa, N.C.A.; Abe, J.M.; Subrahmanian, V.S., Remarks on Annotated Logic, Coleção Documentos, Série:Lógica e Teoria da Ciência n. 8, Instituto de Estudos Avançados, Universidade de São Paulo, São Paulo, Julho, 1991.

[Dougherty, 1995] Dougherty, J.; Kohavi, R.; Sahami, M., *Supervised and Unsupervised Discretization of Continuous Features*, Proceedings of the Twelfth International Conference on Machine Learning, pp. 194-201, Morgan Kaufmann Publishers Inc, Tahoe City, California, 9-12 July, 1995.

[Enembreck, 1999] Enembreck, F.; Ávila, B. C.; Sabourin, R., *Decision Tree-Based Paraconsistent Learning*, Proceedings of XIX International Conference of the Chilean Computer Science Society, pp. 32-44, IEEE Computer Society Press, Talca, Chile, 1999.

[Epstein, 1988] Epstein, I., Teoria da Informação, 2 ed., Série Princípios, Editora Ática, São Paulo, 1988.

[Evans, 1994] Evans, B.; Donnelley, R.R. & Sons Company; Fisher D., *Overcoming Process Delays with Decision Tree Induction*, IEEE Expert, pp. 60-66, February, 1994.

[Facon, 1993] Facon, J., *Processamento e Análise de Imagens*, VI Escola Brasileiro-Argentina de Informática, Julho, 1993.

[Facon, 1996] Facon, Jacques, *Morfologia Matemática: Teoria e Exemplos*, Editora Universitária Champagnat da Pontifícia Universidade Católica do Paraná, Curitiba, 1996.

[Facon, 1998] Facon, J., *Princípios Básicos da Visão por Computador e do Processamento de Imagens*, Comunicação Pessoal, Curitiba, 1998.

[Fayad, 1996] Fayyad, U. M.; Shapiro, G. P.; Smyth, P.; Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI Press / The MIT Press, Massachusetts, 1996.

[Freitas, 1998] Freitas, A. A.; Lavington, S. H., *Mining Very Large Databases with Parallel Processing*, Kluwer, 1998.

[Fu, 1996] Fu, Y., *Discovery of Multiple-Level Rules From Large Databases*, Thesis for the Degree of Doctor of Philosophy, Simon Fraser University, July, 1996.

[Gader, 1994] Gader, P.; Gillies, A.; Hepp, D., *Handwritten Character Recognition*, Digital Image Processing Methods, pp. 223-255, Marcel Dekker Inc., New York, 1994.

[Gader, 1996] Gader, P. D.; Khabou, M. A., *Automatic Feature Generation for Handwritten Digit Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, n. 12, pp. 1256-1261, December, 1996.

[Gall, 1992] Gall, S.I., *Neural Networks and Expert Systems*, The MIT Press, Cambridge, Massachusetts, 1992.

[Han, 1998] Han, J.; Cai, Y.; Cercone, N.,*Concept-Based Data Classification in Relational Databases*, in Workshop Notes of

1991 AAAI Workshop on Knowledge Discovery in Databases (KDD'91), pp. 77-94, Anaheim, CA, July, 1991.

[Han, 1998a] Han, J.; Fu, Y., *Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases*, AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94), pp. 157-168 Seattle, WA, July 1994.

[Hätönen, 1996] Hätönen, K.; Klemettinen, M.; Mannila, H.; Ronkainen, P.; Toivonen, H., *Rule Discovery in Alarm Databases*, Series of Publications C, n° C-1996-7, University of Helsinki, Department of Computer Science, Helsinki, March, 1996.

[Hanson, 1990] Hanson, R.; Stutz, J.; Cheeseman, P., *Bayesian Classification Theory*, Technical Report FIA-90-12-7-01, NASA Ames Research Center, 1990, 9 pg.

[Henkind, 1988] Henkind, S. J.; Harrison, M.C., *An Analysis of Four Uncertainty Calculi*, IEEE Transactions on Systems, Man and Cybernetics, vol. 18, n. 5, September/October, 1988.

[Hermens, 1993] Hermens, L.A.; Schlimmer, J. C., *A Machine-Learning Apprentice for the Completion of Repetitive Forms*, IEEE Expert, pp. 28-33, February, 1994.

[Holsheimer, 1994] Holsheimer, M.; Siebes, A.P.J.M., *Data Mining: the search for knowledge in databases*, Technical Report CS-R9406, Centrum voor Wiskunde en Informatica CWI, Amsterdam, The Netherlands, 1994, 78 pg.

[Hunt, 1966] Hunt, E. B.; Marin, J.; Stone, P.H., *Experiments in Induction*, Academic Press, New York, 1966.

[Janikow, 1998] Janikow, C. Z., *Fuzzy Decision Trees: Issues and Methods*, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, vol. 28, n. 1, February, 1998.

[Jim, 1995] Jim, K.; Lai, J.; Wüthrich, B., *A Data Mining Algorithm Optimal for Single Rules*, The Hong Kong University of Science and Technology, Kowloon, Hong Kong.

[Joachims, 1996] Joachims, T., *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*, Technical Report CMU-CS-96-118, School of Computer Science Carnegie Mellon University, Pittsburg, March, 1996, 24 pg.

[Kamber, 1998] Kamber,M.; Winstone, L.; Gong, W.; Cheng, S.; Han, J., *Generalization and Decision Tree Induction: Efficient Classification in Data Mining*, Proc. of 1997 Int'l Workshop on Research Issues on Data Engineering (RIDE'97), pp. 111-120, Birmingham, England, April, 1997.

[Kerber, 1992] Kerber, R., *ChiMerge: Discretization of Numeric Attributes*, Learning: Inductive, AAAI, pp. 123-128, 1992.

[Kifer, 1989] Kifer, M.; Subrahmanian, V.S., *Theory of Generalized Annotated Logic Programming and its Aplications*, Department of Computer Science SUNY at Stony Brook, Stony Brook, New York, 1989.

[Klopotek, 1997] Klopotek, M.A.; Wierzchón, S.T., *Qualitative Versus Quantitative Interpretation of Mathematical Theory of Evidence*, [in:] Z.W.Ras', A. Skowron Eds: Foundations of Intelligent Systems 7. Proc. ISMIS'97, Charlotte NC, Lecture

Notes in Artificial Intelligence 1325, pp. 391-400, Springer, 15-17 October, 1997.

[Kohavi, 1996] Kohavi, R.; Sahami, M., *Error-Based Discretization of Continuous Features*, Proceedings of the Second International Conference on Knowledge Discovery & Data Mining, pp. 114-119, AAAI Press, California, 1996.

[Kononenko, 1995] Kononenko, I., *On Biases in Estimating Multi-Valued Attributes*, Proc. of Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), vol. 2, pp. 1034-1040, 20-25 August, Montreal, Québec, Canada, 1995.

[Ladeira, 1996] Ladeira, M.; Viccari, R. M., *Representação de Conhecimento Incerto*, XIII Brazilian Symposium on Artificial Intelligence SBIA'96, Curitiba, October, 1996.

[Lee, 1996] Lee, S., *Off-Line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, n. 6, pp. 648-652, June, 1996.

[Lee, 1996a] Lee, L.L.; Berger, T.; Aviczer, E.,*Reliable On-Line Human Signature Verification Systems*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, n. 6, pp. 643-647, June, 1996.

[Lovell, 1996] Lovell, B. C.; Bradley, A. P., *The Multiscale Classifier*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, n. 2, pp. 124-137, February, 1996.

[Luger, 1993] Luger, G. F.; Stubblefield, W., *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Benjamim/Cummings Publishing Company, 2 ed., Redwood City, California, 1993.

[Ma, 1995] Ma, Y., *Inductive Classifier Learning From Data: An Extended Bayesian Belief Function Approach*, Thesis for the degre of Doctor of Philosophy in Computer Science, Graduate College of the University of Illinois at Urbana-Champaign, Urbana, Illinois, 1995.

[Marcus, 1986] Marcus, C.; Arity Corporation, *Prolog Programming: Application for Database Systems, Expert Systems and Natural Language Systems*, Addison-Wesley Publishing Company, USA, 1986.

[Matheron, 1975] Matheron, G., *Random Sets and Integral Geometry*, Wiley and Sons, New York, 1975.

[McCallum, 1998] McCallum, A.; Nigam, K., *A Comparison of Event Models for Naive Bayes Text Classification*, Fifthteenth National Conference on Artificial Intelligence (AAAI-98), 26-30 July, Madison, Wisconsin, 1998.

[McClean, 1998] McClean, S.; Scotney, B.; Shapcott, M., *Aggregation of Inprecise and Uncertain Information for Knowlegde Discovery in Databases*, 4th International Conference on Knowledge Discovery and Data Mining, American Association for Artificial Intelligence, pp. 269-273, New York, 1998.

[Mackay, 1997] Mackay, D.J.C., *Information Theory, Pattern Recognition & Neural Networks*, pp. 01-24, May, 1997.

[Mitchell, 1997] Mitchell, T. M., *Machine Learning*, McGraw-Hill Series in Computer Science, McGraw-Hill Companies, USA, 1997.

[Monard, 1997] Monard, M.C.; Batista, G.E.A.P.A.; Kawamoto, S.; Pugliesi, J.B., *Uma Introdução ao Aprendizado Simbólico de Máquina por Exemplos*, n° 29, Notas Didáticas do Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, São Carlos, 1997.

[Mori, 1992] Mori, S.; Suen, C.Y.; Yamamoto,K., *Historical Review of OCR Research and Development*, Proceeedings of IEEE, vol 80, n. 7, July, 1992.

[Murphy, 1994] Murphy, P.M.; Pazzani, M.J., *Exploring the Decision Forest: An Empirical Investigation of Occam's Razor in Decision Tree Induction*, The Journal of Artificial Intelligence Research, vol. 1, Morgan Kauffman Publishers, San Francisco, California, pp. 257-275, August 1993 - June 1994.

[Murphy, 1995] Murphy, P.M., *UCI repository of machine learning databases — a machine-readable data repository*, Mantained at the Department of Information and Computer Science, University of California, Irvine. Anonymous FTP from ics.uci.edu in the directory pub/machine-learning-databases, 1995.

[Murtagh, 1996] Murtagh, F; Campbell, J., *Image Pocessing (AC460)*, BSc Applied Computing (1220), Year 4, University of Ulster, Magee College, pp. 1-16, October, 1996.

[Murtagh, 1996a] Murtagh, F; Campbell, J., *Image Pocessing (AC460)*, Chapters 8-9, BSc Applied Computing (1220), Year 4, University of Ulster, Magee College, November 1996.

[Nadler, 1993] Nadler, M.; Smith, E.P., *Pattern Recognition Engineering*, A Wiley-Interscience Publication, USA, 1993.

[Neapolitan, 1990] Neapolitan, R. E., *Probabilistic Reasoning In Expert Systems: Theory and Algorithms*, Wiley-Interscience Publication, USA, 1990.

[Nicoletti, 1993] Nicoletti, M. C.; Monard M. C., *Herbrand Interpretation, Model and Least Model within the Framework of Logic Programming*, Notas do Instituto de Ciências Matemáticas de São Carlos, ISSN 0103-2577, São Carlos, Junho, 1993, 30 pg.

[Ng, 1990] NG, Keung-Chi; Abramson, B., *Uncertainty Management in Expert Systems*, IEEE Expert, pp. 29-47, April, 1990.

[Ng, 1991] Ng, R.; Subrahmanian, V.S., *Relating Dempster-Shafer Theory to Stable Semantics*, Technical Report, UMIACS-TR-91-49, CS-TR-2647, April, 1991, 40 pg.

[Ng, 1992] Ng, R.; Subrahmanian, V.S., *Probabilistic Logic Programming*, vol 101, n. 2, pp. 150-201, Academic Press, Belgium, December 1992.

[Pearl, 1988] Pearl, J., *Probabilistic Reasoning in Intelligent Systems,: Networks of Plausible Inference*, 2 ed., Morgan Kaufmann Publishers, San Mateo, California, 1988.

[Pfahringer, 1995] Pfahringer, B., *Compression-Based Discretization of Continuous Attributes*, Proceedings of the Twelfth International Conference on Machine Learning, pp. 456-463, Morgan Kaufmann Publishers Inc, Tahoe City, California, 9-12 July, 1995.

[Pitas, 1992] Pitas, I.; Milios, E.; Venetsanopoulos, A.N., *A Minimum Entropy Approach to Rule Learning from Examples*, IEEE Transactions on Systems, Man and Cybernetics, vol. 22, n. 4, pp. 621-635, July/August, 1992.

[Quinlan, 1986] Quinlan, J.R., *Induction of Decision Trees*, Machine Learning, vol. 1, Kluwer Academic Publishers, pp. 81-106, Netherlands, 1986.

[Quinlan, 1993] Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kauffman Publishers, San Mateo, California, 1993.

[Rich, 1994] Rich, E.; Knight, K., *Inteligência Artificial*, 2 ed., Makron Books do Brasil Editora Ltda, São Paulo, 1994.

[Richeldi, 1995] Richeldi, M.; Rossotto, M., *Class-Driven Statistical Discretization of Continuous Attributes (Extended Abstract)*, 8th ECML, pp. 335-338, Heraclion, Creete, Greece, April, 1995.

[Russel, 1995] Russel, S.; Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[Sabourin, 1993] Sabourin, R.; Cheriet, M.; Genest, G., *An Extended-Shadow-Code Based Approach for Off-Line Signature Ve-*

*rification*, Proceedings of the Second International Conference on Document Analysis and Recognition, Tsukuba Science City, Japan, 20-22 October, 1993.

[Sabourin, 1994] Sabourin, R.; Genest, G., *An Extended-Shadow-Code Based Approach for Off-Line Signature Verification: Part I - Evaluation of the Bar Mask definition*, Proceeding of the 12 th IAPR International Conference on Pattern Recognition, Jerusalem, Israel, 9-13 October, 1994.

[Sabourin, 1997] Sabourin, R.; Genest, G.; Prêteux F.J.,*Off-Line Signature Verification by Local Granulometric Size Distribuitions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, n. 9, September, 1997.

[Sandri, 1996] Sandri, S., *Numerical Models for Treatment of Imperfect Information in Knowledge-Based Systems*, Proceedings of 13th Brazilian Symposium on Artificial Intelligence SBIA'96, Curitiba, Brazil, 1996.

[Sandri, 1997] Sandri, S., *Introdução à Lógica Fuzzy*, Minicurso Apresentado no 3° Simpósio Brasileiro de Automação Inteligente, UFES, Vitória, 3-5 Setembro, 1997.

[Sasisekharan, 1996] Sasisekharan, R.; Seshadri, V.; Weiss, S.M., *Data Mining and Forescating in Large-Scale Telecommunications Networks*, IEEE Expert, vol. 11, n. 1, pp. 37-43, February, 1996.

[Schalkoff, 1992] Schalkoff, R., *Pattrern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, Inc, USA, 1992.

[Senior, 1992] Senior, A.W., *Off-Line Handwriting Recognition: A Review and Experiments*, Technical Report CUED/F-INFENG/TR 105, 23 pg., Cambridge, England, December 1992.

[Shohan, 1994] Shohan, Y, *Artificial Intelligence: Tecniques in Prolog*, Morgan Kaufmann Publishers, San Francisco, California, 1994.

[Sterling, 1986] Sterling, L.; Shapiro, E., *The Art of Prolog: Advanced Programming Techniques*, The MIT Press, Massachussets, 1986.

[Subrahmanian, 1987a] Subrahmanian, V.S.; *On the Semantics of Quantitative Logic Programs*, Proceedings of 4th IEEE Symposium on Logic Programming, San Francisco, September, pp. 173-182, 1987.

[Subrahmanian, 1987b] Subrahmanian, V.S., *Towards a Theory of Evidential Reasoning in Logic Programming*, Logic Colloquium '87, The European Summer Meeting of the Association for Symbolic Logic, Granada, Spain, July, 1987.

[Tanomaru, 1995] Tanomaru, J., *Motivação, Fundamentos e Aplicações de Algoritmos Genéticos*, II Congresso Brasileiro de Redes Neurais III Escola de Redes Neurais, Curitiba, 29 Outubro-01 Novembro, 1995.

[van Emden, 1986] van Emden, M. H., *Quantitative Deduction and Its Fixpoint Theory*, The Jornal of Logic Programming, vol. 1, pp. 37-53, New York, 1986.

[Wang, 1998] Wang, H.; Düntsch, I.; Bell, D., *Data Reduction Based on Hyper Relations*, School of Information and Software Engineering, University of Ulster, Newtownabbey, N. Ireland, 1998.

[Weeks, 1996] Weeks, A. R., *Fundamentals of Electronic Image Processing*, SPIE/IEEE series on image science & engeneering, The Society Photo-Optical Instrumentation Engineers, USA, 1996.

[Wolkenhauer, 1997] Wolkenhauer, O., *Qualitative Uncertainty Models from Random Set Theory*, Advances in Intelligent Data Analysis (IDA-97), Springer-Verlag, pp. 609-620, Berlin, Heidelberg, 1997.

[Wüthrich, 1997] Wüthrich, B., *Discovering Probabilistic Decision Rules*, The Hong Kong University of Science and Technology, Kowloon, Hong Kong, 1997.

[Yu, 1997] Yu, H.; Ramer, A., *A Combined Approach to Uncertain Data Analysis*, Advances in Intelligent Data Analysis (IDA-97), Springer-Verlag, pp. 123-134, Berlin, Heidelberg, 1997.

[Zadeh, 1965] Zadeh, L. A., *Fuzzy Sets*, Information and Control, 8, pp. 338-353, 1965.

[Zhou, 1995] Zhou, X.M.; Dillon, T.S., *Theoretical and Practical Considerations of Uncertainty and Complexity in Automated Knowledge Acquisition*, IEEE Transactions on Knowledge and Data Engineering, vol. 7, n. 5, pp. 699-712, October, 1995.

# Apêndice A

# Introdução ao Aprendizado de Máquina

## A.1 Considerações Iniciais

Um sistema de AM, geralmente, é formado de elementos básicos. Dentre estes elementos está a base de dados, que é constituída por um conjunto de exemplos de treinamento, ou observações, $E = \{e_1, ..., e_n\}$. Outros elementos do sistema são um ou mais algoritmos de AM, que, se aplicados sobre $E$, são capazes de gerar uma base de conhecimento $BC$ formada por vários conceitos, ou uma outra representação, como por exemplo redes neurais — através da qual inferências podem ser realizadas. Nesta Seção serão fornecidos os conceitos básicos envolvidos na tarefa de Aprendizado de Máquina.

## A.2 Aspectos Gerais

O formato dos exemplos de treinamento influencia a estratégia de aprendizado a ser aplicada. Considere-se um sistema que possui uma base de dados $E = \{e_1, ..., e_n\}$ de observações, onde cada $e_i$ apresenta o seguinte formato:

$$e_i =< a_1, ..., a_m >$$

O conjunto $\{a_1, ..., a_{m-1}\}$ é formado pelos *atributos previsores* e $a_m$ é o *atributo meta*. Pode-se dizer que a estratégia mais adequada de aprendizado a ser aplicada neste caso é o aprendizado supervisionado. Essa forte suposição é assumida devido à presença do atributo meta $a_m$. Isso indica que para cada observação presente em $E$ um supervisor — ou professor — definiu a classe correspondente, caracterizando o aprendizado supervisionado. Por outro lado, quando não se dispõe de um professor, e, portanto, de um atributo meta, diz-se que a estratégia de aprendizado a ser empregada é o aprendizado não-supervisionado. É importante ressaltar que técnicas de aprendizado não-supervisionado podem ser aplicadas em bases de dados como a descrita anteriormente.

Prosseguindo a descrição do formato dos exemplos que formam $E$, pode-se dizer que $o = m - 1$ e $A = \{a_1, ..., a_o\}$ é o conjunto de atributos previsores. Cada atributo $a_j$ está associado a um domínio $d_j$ que representa o conjunto de valores possíveis para o atributo $a_j$. Portanto, pode-se dizer que existe um conjunto $D = \{d_1, ..., d_m\}$ que define todos os domínios de $E$. O conjunto $d_m$ representa as hipóteses mutuamente exclusivas que podem estar associadas a um exemplo de $E$.

A partir das definições anteriores, pode-se definir o espaço de tuplas $T$ de $E$ da seguinte forma:

$$T_E = \prod_{i=1}^{m} d_i \tag{A.1}$$

onde, $E \subseteq T_E$.

A tarefa do algoritmo de AM pode consistir, portanto, em percorrer o espaço de tuplas $T$, utilizar determinadas operações sobre os conceitos gerados e escolher, segundo algum critério predefinido, quais os melhores conceitos ou descrições. Obviamente, nem sempre o espaço $T$ necessita ser percorrido inteiramente para que sejam descobertas as descrições, uma vez que dificilmente $|E| = |T_E|$, assumindo a não existência de duplicidade em $E$. Além disso, cada descrição gerada representa um subconjunto de $E$, o que diminui acentuadamente o espaço restante a ser percorrido durante o processo de busca.

A seguir, é apresentado um exemplo — Exemplo A.2.1 — que utiliza os conceitos introduzidos.

**Exemplo A.2.1** Conceitos utilizados em Aprendizado de Máquina.

$E$

| $i$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $\mathbf{a_5}$ |
|---|---|---|---|---|---|
| $e_1$ | 1 | 1 | 1 | 3 | ***1*** |
| $e_2$ | *5* | *2* | *10* | *2* | ***1*** |
| $e_3$ | *3* | *2* | *10* | *5* | ***1*** |
| $e_4$ | *2* | *1* | *1* | *3* | ***2*** |
| $e_5$ | *3* | *2* | *10* | *2* | ***3*** |
| $e_6$ | *4* | *2* | *10* | *5* | ***1*** |

$D$

| $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ |
|---|---|---|---|---|
| $\{1,2,3,4,5\}$ | $\{1,2\}$ | $\{1,10\}$ | $\{2,3,5\}$ | $\{1,2,3\}$ |

$A = \{a_1, a_2, a_3, a_4, a_5\}$

$E = \{e_1, ..., e_6\}$

$T_E = \{d_1 \times d_2 \times d_3 \times d_4 \times d_5\}$

$|E| = 6$

$|T_E| = 5 \times 2 \times 2 \times 3 \times 3 \ = 180$

$D = \{d_1, d_2, d_3, d_4, d_5\}$

onde, $E$ é o conjunto de exemplos, $T_E$ é o espaço de tuplas de $E$, $A$ é o conjunto de atributos, $D$ é o conjunto de domínios para os atributos, $d_i$ é o domínio do atributo $i$ e $a_5$ é o atributo meta.

## A.3 Aprendizado como Problema de Busca

Na Seção anterior foi introduzido o conceito de busca por descrições. Agora, este processo será abordado mais profundamente.

Geralmente, um algoritmo de aprendizado começa o processamento com uma descrição inicial. A partir da qual, novas descrições são geradas aplicando-se determinadas operações. Basicamente, duas operações podem ser realizadas: generalização e/ou especialização.

A primeira operação — generalização — sobre uma descrição $D$ produz uma descrição $D\prime$ que cobre um superconjunto de exemplos cobertos por $D$. Porém, isso não significa, necessariamente, que $D\prime$ cubra mais exemplos corretamente.

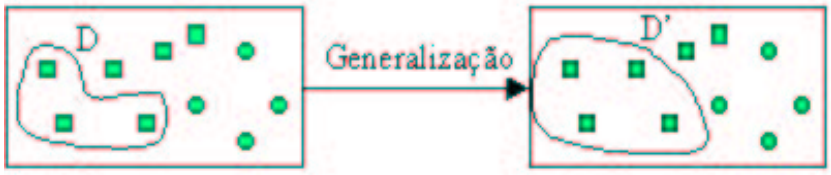



Figura A.1: Generalização.

A Figura A.1 apresenta uma operação de generalização que gera uma descrição realmente melhor do que a original. Porém, na Figura A.2, uma determinada medida de avaliação necessita ser tomada para se verificar a validade desta operação, uma vez que exemplos negativos (representados por círculos) — que não pertencem à hipótese associada à descrição — são classificados positivamente. Basicamente, a

operação de generalização consiste em eliminar testes presentes no antecedente da regra que forma uma descrição, ou aumentar o intervalo de valores possíveis nesses testes.

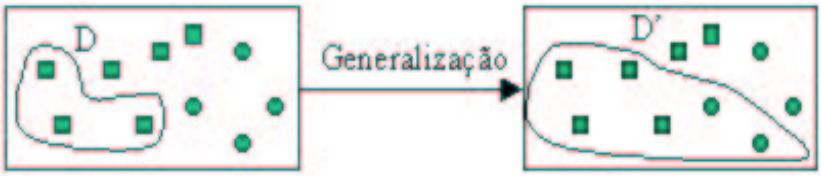



Figura A.2: Operação de Generalização com Exemplos Incorretos.

A segunda operação — especialização — consiste em produzir, a partir de uma descrição $D$, uma outra descrição $D\prime$, que cobre um subconjunto de elementos cobertos por $D$. A Figura A.3 apresenta a especialização de $D$ em $D\prime$. A operação de especialização apresenta características inversas em relação à generalização. Portanto, na prática, a especialização de uma descrição consiste em adicionar novos testes ao antecedente da regra que a forma, ou diminuir o intervalo de valores possíveis para esses testes.

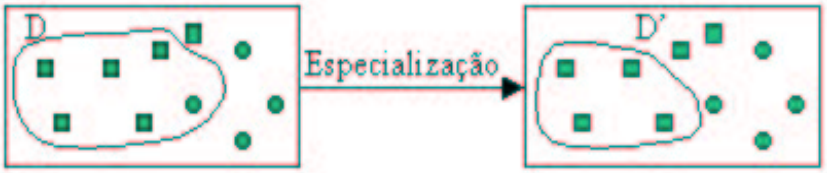



Figura A.3: Operação de Especialização.

Existem diversas formas de se representar uma descrição. A forma mais comumente utilizada, é através de regras com o seguinte formato:

**se** <*condição*> ***então*** <*hip{ótese*>

Os testes que formam o antecedente da regra — <*condição*> — são realizados sobre os valores dos atributos e necessitam de informações adicionais sobre os

respectivos domínios. Os domínios dos atributos podem ser caracterizados como pertencentes aos seguintes três tipos:

1. *Nominal ou categórico*: os valores possíveis são formados por símbolos ou rótulos independentes, ou seja, não estão ordenados;

2. *Linear*: os elementos do domínio estão totalmente ordenados. Os valores podem ser numéricos ($\{1, ..., 10\}$) ou simbólicos ($\{pequeno, médio, grande\}$). Atributos lineares podem, ainda, ser discretos (número limitado de valores) ou contínuos (número infinito de valores possíveis);

3. *Parcialmente Ordenado*: existe uma ordem parcial entre os elementos, o que torna possível a criação de uma hierarquia onde há um valor que representa todo o domínio (pai de todos) e valores filhos que representam subhierarquias.

Durante a descrição das operações básicas de generalização e especialização verificou-se a necessidade de um critério de avaliação, com o objetivo de estabelecer uma ordenação das descrições em termos de qualidade. Inúmeras formas de avaliação de regras foram criadas, muitas delas inerentes a novos métodos de raciocínio desenvolvidos [Cho, 1998]. Porém, algumas formas básicas de avaliação estatística continuam sendo utilizadas como referência para novos trabalhos. Para que se possa definir tais critérios, a seguinte notação será utilizada:

- $R{:}C$: é uma regra que associa exemplos à classe $C$ — $R$ corresponde ao antecedente da regra, enquanto $C$ representa o conseqüente da regra;

- $E$: é a base de dados;

- $PPositivos$: subconjunto de $E$ de exemplos cobertos por $R$ e por $C$;

- $Positivos$: subconjunto de $E$ de exemplos cobertos por $R$.

As medidas mais comuns para avaliação de uma regra $R$:$C$ são as seguintes:

- *Confidência*: mede a precisão de uma regra. Uma regra $R$:$C$ possui confidência de 100% se todos os exemplos cobertos são $PPositivos$. Pode-se calcular a confidência de R da seguinte forma:

$$Conf_{R:C} = \frac{|PPositivos|}{|Positivos|} \times 100 \tag{A.2}$$

- *Suporte*: mede a importância de uma regra em relação à base de dados. Geralmente, uma regra representa um subconjunto da base de dados. Uma regra pode ser mais significante, em relação a outras, se ela representa grande parte da base de dados. Essa medida é dada a seguir:

$$Sup_{R:C} = \frac{|PPositivos|}{|E|} \tag{A.3}$$

Além das medidas básicas apresentadas em [Cho, 1998], o autor mostra que é possível combiná-las para se obter novas formas de avaliação.

Na Figura A.4 são mostrados, de forma ilustrativa, os possíveis comportamentos para uma determinada descrição $R : +$[1] , em relação à classificação de exemplos, onde "+" e "−" representam exemplos pertencentes às classes "+" e "−", respectivamente [Monard, 1997].

[1] $R : +$ representa uma regra com antecessor $R$ e que cobre exemplos da classe +

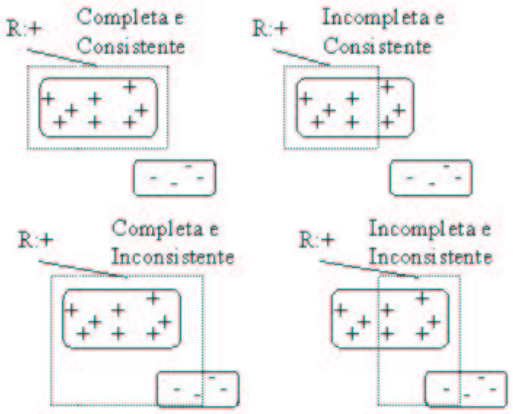



Figura A.4: Comportamentos de uma Descrição.

# Apêndice B

# Incerteza

## B.1 Exemplo de Aplicação da Teoria da Probabilidade

A seguir é apresentado um pequeno exemplo da aplicação da Teoria da Probabilidade.

O problema consiste em identificar a doença $H \in \{H_1, H_2, H_3\}$ mais provável, dados os sintomas pertencentes ao conjunto $E = \{E_1, E_2\}$ e demais probabilidades.

Dado:

| $i$ | 1 | 2 | 3 |
|---|---|---|---|
| $p(H_i)$ | 0.5 | 0.3 | 0.2 |
| $p(E_1\|H_i)$ | 0.4 | 0.8 | 0.3 |
| $p(E_2\|H_i)$ | 0.7 | 0.9 | 0.0 |

$$p(E_1E_2|H_1) = 0.4 \times 0.7 \times 0.5$$

$$p(E_1E_2|H_2) = 0.8 \times 0.9 \times 0.3$$

$$
\begin{aligned}
p(E_1E_2|H_3) &= 0.3 \times 0.0 \times 0.2 \\
p(H_1|E_1E_2) &= \frac{0.4 \times 0.7 \times 0.5}{(0.4 \times 0.7 \times 0.5) \times (0.8 \times 0.9 \times 0.3) \times (0.3 \times 0.0 \times 0.2)} = 0.393 \\
p(H_2|E_1E_2) &= \frac{0.8 \times 0.9 \times 0.3}{(0.4 \times 0.7 \times 0.5) \times (0.8 \times 0.9 \times 0.3) \times (0.3 \times 0.0 \times 0.2)} = 0.607 \\
p(H_3|E_1E_2) &= \frac{0.3 \times 0.0 \times 0.2}{(0.4 \times 0.7 \times 0.5) \times (0.8 \times 0.9 \times 0.3) \times (0.3 \times 0.0 \times 0.2)} = 0.0
\end{aligned}
$$

Baseando-se nos valores probabilísticos obtidos, pode-se concluir que um paciente que apresente os dois sintomas $E_1$ e $E_2$ terá uma maior probabilidade de portar a doença representada pela hipótese $H_2$.

## B.2 Exemplo de Aplicação da Teoria de Dempster-Shafer

A seguir é fornecido um exemplo retirado de [Rich, 1994] e [Ladeira, 1996] que apresenta a utilização da DST em um pequeno sistema de diagnóstico.

Suponha-se que se deseja diagnosticar a doença de um determinado paciente e sabe-se, com certeza, que a hipótese correta pertence a $\theta = \{Alergia = Al, Gripe = Gr, Resfriado = Re, Pneumonia = Pn\}$, considerando as hipóteses exaustiva e mutuamente exclusivas.

A situação inicial é de total ignorância, ou seja, não há como definir subconjuntos iniciais de $\theta$, logo, $m(\theta) = 1$. Suponha-se, porém, que de acordo com a opinião médica, exista uma evidência de 0.6 indicando que o diagnóstico correto está no conjunto $A = \{Gr, Re, Pn\}$. O médico pode ser induzido a obter esta decisão, por exemplo, verificando a evidência de febre no paciente. Portanto, agora, tem-se:

$$m(A) = 0.6 \qquad \text{e} \qquad m(\theta) = 0.4$$

É importante notar que o complemento da crença atribuída ao conjunto $A$ foi

associado ao conjunto total de hipóteses de $\theta$, e não apenas ao complemento de $A$. Outra característica encontrada em DST é que outras evidências podem atribuir crenças diferentes ao mesmo conjunto de hipóteses. Portanto, é necessária uma fórmula para a combinação das diversas crenças associadas a uma determinada hipótese, de acordo com as evidências existentes.

A regra de combinação de DST que combina duas funções de crença $m_1$ e $m_2$ sobre o conjunto $X$, que representa os subconjuntos de $\theta$ onde a função $m_1$ atribui valores de crença diferentes de zero, e sobre o conjunto $Y$, que representa os subconjuntos de $\theta$ onde a função $m_2$ atribui valores de crenças diferentes de zero, é a seguinte:

$$m_3(Z) = \frac{\sum_{Z=X\cap Y} m_1(X) \times m_2(Y)}{1 - \sum_{X\cap Y=\phi} m1(X) \times m_2(Y)} \tag{B.1}$$

onde, $\phi$ é o conjunto vazio.

Assumindo que no exemplo a função $m$ representa a função $m_1$ e que a função $m_2$ é obtida após a observação da evidência de nariz escorrendo, tem-se:

$$m1(\{Gr, Re, Pn\}) = 0.6$$

$$m1(\theta) = 0.4$$

e

$$m2(\{Gr, Re, Al\}) = 0.8$$

$$m2(\theta) = 0.2$$

Através da equação de combinação, pode-se obter a função $m_3$ gerada a partir da combinação entre as funções $m_1$ e $m_2$ — Tabela B.1.

Estendendo o número de evidências existentes, pode-se adicionar a função básica $m_4$, que representa o desaparecimento da doença do paciente quando o mesmo viaja, da seguinte forma:

| $m_1$ \ $m_2$ | | {Al,Gr,Re} 0.80 | | $\theta$ | 0.20 |
|---|---|---|---|---|---|
| {Gr,Re,Pn} | 0.60 | {Gr,Re} | 0.48 | {Gr,Re,Pn} | 0.12 |
| $\theta$ | 0.40 | {Al,Gr,Re} | 0.32 | $\theta$ | 0.08 |

Tabela B.1: Combinação de $m_1$ e $m_2$.

| $m_3$ \ $m_4$ | | {Al} 0.900 | | $\theta$ | 0.100 |
|---|---|---|---|---|---|
| {Gr,Re} | 0.48 | $\phi$ | 0.432 | {Gr,Re} | 0.048 |
| {Al,Gr,Re} | 0.32 | {Al} | 0.288 | {Al,Gr,Re} | 0.032 |
| {Gr,Re,Pn} | 0.12 | $\phi$ | 0.108 | {Gr,Re,Pn} | 0.012 |
| $\theta$ | 0.08 | {Al} | 0.072 | $\theta$ | 0.008 |

Tabela B.2: Combinação de $m_3$ e $m_4$ — função $m_5$.

$$m_4(\{Al\}) = 0.90$$

$$m_4(\theta) = 0.10$$

Pode-se, portanto, combinar as funções $m_3$ e $m_4$ com o objetivo de se obter a função $m_5$, como ilustrado na Tabela B.2

Há diferenças na forma de realizar combinações para a obtenção das funções $m_3$ e $m_5$. Nas combinações da função $m_3$, pode-se perceber que não há ocorrência de $\phi$ (conjunto vazio) nas interseções entre os subconjuntos de hipóteses. Este fato faz com que o denominador da equação de combinação obtenha sempre o valor 1, atribuindo ao cálculo das crenças o valor obtido apenas pela multiplicação das crenças dos subconjuntos que estão sendo combinados. Porém, a tabela da função $m_5$ apresenta crença total de 0.54 no conjunto vazio, uma vez que todas as crenças do mesmo subconjunto são somadas (0.432+0.108). Isto torna necessário que os demais valores obtidos a partir de subconjuntos diferentes de vazio sejam normalizados pelo valor 0.46 $(1 - 0.54)$, que representa a parte da probabilidade onde a solução correta para o problema se encontra. De maneira semelhante ao subconjunto vazio, o subconjunto $\{Al\}$ também ocorre mais de uma vez, logo a crença em $\{Al\}$ também

| | {Al} | {Gr,Re} | {Al,Gr,Re} | {Gr,Re,Pn} | $\theta$ |
|---|---|---|---|---|---|
| $m_5$ | 0.783 | 0.104 | 0.070 | 0.026 | 0.017 |
| Crença | 0.783 | 0.104 | 0.957 | 0.130 | 1.0 |
| Plausab. | 0.870 | 0.217 | 1 | 0.217 | 1.0 |

Tabela B.3: Valores Normalizados de $m_5$ e seus Valores de *Crença* e *Plausibilidade*.

necessita ser somada $(0.288 + 0.072)$.

O processo de normalização citado anteriormente é simples; basta dividir os valores das crenças de cada subconjunto diferente de vazio pelo complemento da crença no conjunto vazio (0.46). A Tabela B.3 apresenta os valores normalizados de $m_5$ e seus intervalos de Crença e Plausibilidade.

O cálculo da crença de um determinado subconjunto é trivial. Porém, o cálculo da Plausibilidade pode ser obtido, diretamente, somando-se os valores de $m$ para os subconjuntos cuja interseção não é vazia, como pode ser observado na Tabela B.2.

# Apêndice C

# Morfologia Matemática Binária

Morfologia Matemática é o ramo da ciência que estuda a estrutura geométrica de objetos presentes em uma imagem. Os primeiros estudos sobre Morfologia Matemática foram desenvolvidos por Georges Matheron e Jean Serra [Matheron, 1975], com o objetivo de realizar operações sobre imagens [Facon, 1993] [Facon, 1996], como por exemplo, realce, análise das formas, compressão, etc.

Toda operação morfológica utiliza um elemento estruturante [Facon, 1998] — ES — que consiste de um conjunto de pixels de tamanho e forma definidos, que é comparado, de forma específica para cada operação, com o conjunto completo de pixels da imagem original. A imagem resultante, portanto, é obtida através de todas essas comparações.

Seguindo a notação utilizada em [Facon, 1996], um elemento estruturante é denotado por $B$ e $B_x$ representa o elemento estruturante $B$ com elemento central $x$. Os pixels que formam as imagens e os elementos estruturantes podem ser de dois tipos:

"." denota um pixel inativo;

"•" denota um pixel ativo, ou seja, que interage com o conjunto $X$ de pixels da

imagem.

A interação entre os pixels ativos de $B$ e $X$, produz um pixel ativo em $X$, na posição apontada pelo elemento central de $B$. Caso esta interação não seja possível, o pixel em $X$ apontado pelo elemento central de $B$ será inativo.

A seguir é ilustrada uma representação gráfica de $B$, onde o pixel entre "()" representa o elemento central:

$$B = \left\{ \begin{array}{ccc} \cdot & \bullet & \cdot \\ \bullet & (\cdot) & \bullet \\ \cdot & \bullet & \cdot \end{array} \right\}$$

A não utilização de "()" para denotar o elemento central, indica que o mesmo é representado pelo centro físico de $B$.

De forma semelhante à representação de $B$, uma imagem $X$ é representada por um conjunto de pixels entre "[]" da seguinte maneira:

$$X = \left[ \begin{array}{ccccc} \bullet & \bullet & \cdot & \cdot & \cdot \\ \bullet & \bullet & \cdot & \cdot & \cdot \\ \bullet & \bullet & \bullet & \bullet & \cdot \\ \bullet & \bullet & \bullet & \bullet & \cdot \\ \cdot & \cdot & \bullet & \bullet & \cdot \end{array} \right]$$

O elemento estruturante transposto de $B$, denotado $\tilde{B}$, é obtido da seguinte forma:

$$Se\ B = \left\{ \begin{array}{ccc} \cdot & \bullet & \cdot \\ \cdot & (\cdot) & \bullet \\ \cdot & \cdot & \cdot \end{array} \right\}\ então\ \tilde{B} = \left\{ \begin{array}{ccc} \cdot & (\cdot) & \cdot \\ \bullet & \bullet & \cdot \end{array} \right\}$$

## C.1 Translação e Simetria

É possível modificar um subconjunto $B$ de $\mathcal{E}$, por exemplo, através da translação de um vetor $h$ da seguinte forma:

$$B + h = y \in \mathcal{E} \rightarrow y - h \in B$$

O conjunto obtido a partir da translação do vetor $h$ em $B$, $B_h$, pode ser definido, também, da seguinte maneira:

$$y \in \mathcal{E} \rightarrow y + h \in B_h$$

e satisfaz as seguinte propriedades:

$$B_o = B$$

$$(B_h)k = B_{h+k} = (B_k)h$$

Segundo Facon [Facon, 1996], o conjunto central obtido a partir de $B$, por simetria central pela origem ($o$) do sistema de referência, denotado $\tilde{B}$, e chamado de $B$ transposto, é:

$$y \in B \rightarrow -y \in \tilde{B}$$

$$B_h \rightarrow (\tilde{B}) - h$$

## C.2 Operações de Minkowski

Minkowski [Facon, 1996] definiu operações que são utilizadas por muitos autores na definição de operadores morfológicos. Elas são conhecidas como adição e subtração de Minkowski.

Dados dois subconjuntos $A$ e $B$ de $\mathcal{E}$, a adição de Minkowski entre $A$ e $B$ é dada por:

$$A \oplus B = \{x \in \mathcal{E} | \exists a \in A \;\; e \;\; \exists b \in B | x = a + b\} = \cup_{b \in B} A_b$$

De maneira semelhante à adição, a subtração de Minkowski entre $A$ e $B$ é definida como:

$$A \ominus B = \{x \in \mathcal{E} | \forall b \in B, \;\; \exists a \in A | x = a - b\} = \cap_{b \in B} A_b$$

## C.3 Operações Morfológicas Básicas

Nesta Seção serão definidas as duas operações morfológicas básicas — *erosão* e *dilatação* — e, posteriormente, duas operações compostas — *abertura* e *fechamento* — que utilizam as operações básicas.

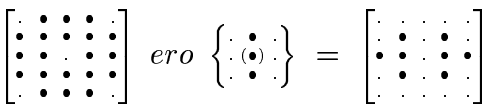
### C.3.1 Operação Erosão

**Definição C.3.1** A erosão de um conjunto $X$ por um elemento estruturante $B$, denotada $X\ ero\ B$, é dada por:

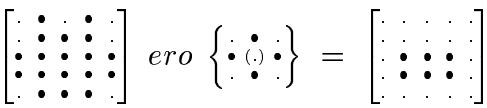
$$X\ ero\ B = \{x \in \mathcal{E} | B_x \subset X\}$$

Na prática, a definição anterior indica que o ES $B$ deve deslizar sobre a imagem $X$, posicionando o elemento central de $B$ — $x$ —, sobre cada pixel de $X$. Em cada posicionamento de $x$, verifica-se a vizinhança de $x$ em $X$ e em $B$. Caso todos os pontos ativos de $B$ também estejam ativos nas respectivas posições em $X$, a posição apontada por $x$ em $X$ se transformará em um pixel ativo, caso contrário, será um pixel desativado. A seguir são apresentados alguns exemplos de operações de erosão.

**Exemplo C.3.1** $X\ ero\ B$.



**Exemplo C.3.2** $X\ ero\ B$.



Segundo Facon, foi demonstrado que a definição de erosão pode estar relacionada à subtração de Minkowski (Seção C.2) da seguinte forma:

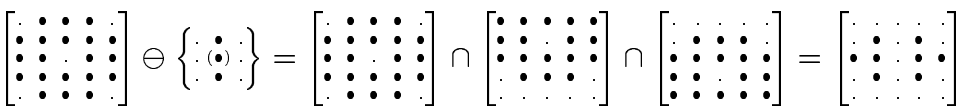
**Definição C.3.2** A erosão de $X$ por $B$ pode ser dada através de:

$$X \; ero \; B = X \ominus \tilde{B} = \cap_{b \in \tilde{B}} X_b = \cap_{b \in B} X_{-b}$$

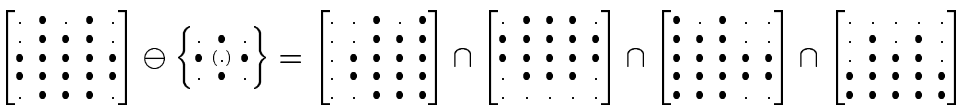
Segundo esta última definição, $X$ é deslocado de acordo com as posições permitidas pelo elemento estruturante $\tilde{B}$. Portanto, agora tem-se que transladar $X$ e não $B$.

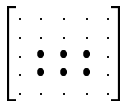
As respectivas soluções dos exemplos anteriores através da subtração de Minkowski são obtidas da seguinte forma:

**Exemplo C.3.3** $X \ominus \tilde{B}$.



**Exemplo C.3.4** $X \ominus \tilde{B}$.





Pode-se dizer que os efeitos obtidos a partir da operação de erosão são:

- *diminuir elementos da imagem*;
- *eliminar elementos de tamanho inferior ao ES*;
- *buracos na imagem são aumentados*;
- *permite a separação de elementos que estão próximos na imagem.*

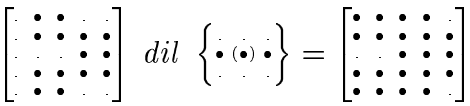
### C.3.2 Operação Dilatação

**Definição C.3.3** A dilatação de um conjunto X com o elemento estruturante B, denotada $X\ dil\ B$, é definida da seguinte forma:

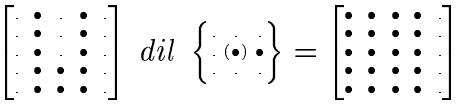
$$X\ dil\ B = \{x \in X | B_x \cap X \neq 0\}$$

De forma semelhante à erosão, o pixel central de $B$ é alinhado com todos os pixels de $X$, diferindo na forma de comparação. Um pixel de $X$ alinhado com $B_x$ será um pixel ativo se existe pelo menos um pixel ativo na sua vizinhança cujo correspondente em $B$ também está ativo. Ou seja, a interseção entre os pixels ativos de $B$ e $X$ é diferente de vazio. A seguir são apresentados alguns exemplos de operações de dilatação.

**Exemplo C.3.5** $X\ dil\ B$.



**Exemplo C.3.6** $X\ dil\ B$.



Uma segunda definição pode ser atribuída à operação de dilatação, utilizando a operação de adição de Minkowski.

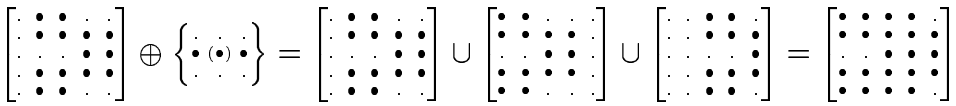
**Definição C.3.4** A dilatação de um conjunto $X$ por um elemento estruturante $B$ pode ser obtida através de:

$$X\ dil\ B = X \oplus \tilde{B} = \cup_{b \in \tilde{B}} X_b = \cup_{b \in B} X_{-b}$$

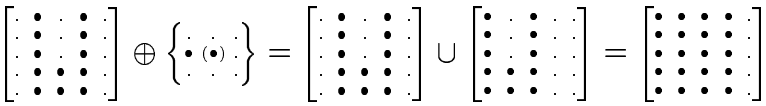
De maneira semelhante à segunda definição de erosão, a última definição de dilatação indica que $X$ é transladado em relação às posições válidas em $B$.

Os exemplos de dilatação anteriores, resolvidos a partir da segunda definição, são ilustrados a seguir.

**Exemplo C.3.7** $X \oplus \tilde{B}$.



**Exemplo C.3.8** $X \oplus \tilde{B}$.



Pode-se encontrar os seguintes efeitos através da operação de dilatação:

- *os elementos da imagem são "engordados";*
- *buracos menores que o ES são preenchidos*;
- *elementos próximos na imagem são conectados.*

### C.3.3 Operações de Abertura e Fechamento Binários

As operações morfológicas estudadas até agora — erosão e dilatação — são capazes de modificar uma imagem original e eliminar alguns defeitos como buracos, conecção entre elementos da imagem, etc. Porém, essas operações deformam a imagem original, uma vez que erosão produz uma imagem menor e a dilatação produz uma imagem maior.

É possível reproduzir os efeitos das operações de erosão e dilatação utilizando-se de outras duas operações morfológicas — abertura e fechamento — sem alterar o tamanho original da imagem.

### Operação de Abertura

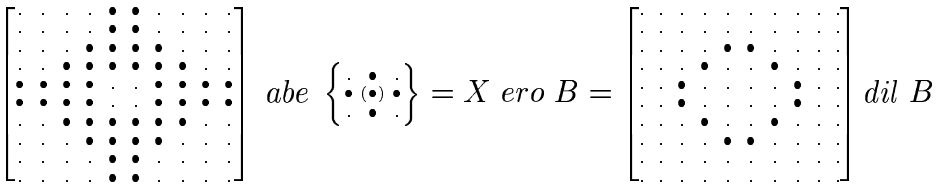
**Definição C.3.5** A abertura de um conjunto $X$ pelo elemento estruturante $B$, denotada $X\ abe\ B$, é dada por:

$$X\ abe\ B = (X\ ero\ B)\ dil\ B = (X \ominus \tilde{B}) \oplus \tilde{B}$$

Como pode-se perceber através da definição anterior, a abertura consiste em aplicar a operação de erosão seguida pela operação de dilatação com o mesmo elemento estruturante.

A seguir são apresentados exemplos de operações de abertura.

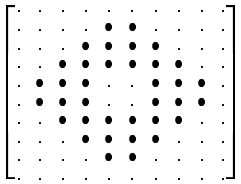
**Exemplo C.3.9** $X\ abe\ B$.



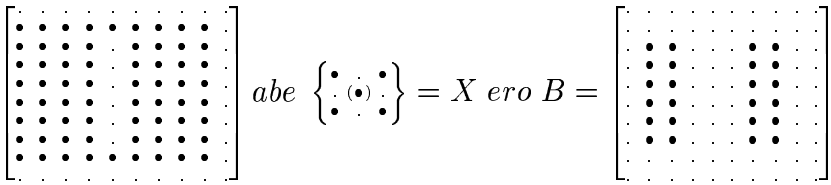
$$= X\ ero\ B =$$



$dil\ B$



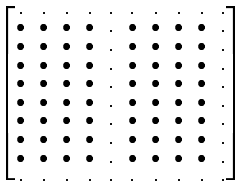
**Exemplo C.3.10** $X\ abe\ B$.



$$= X\ ero\ B =$$



$dil\ B$



Pode-se observar, através da operação de abertura, os seguintes efeitos:

- *geralmente, não retorna a forma original da imagem;*
- *nivela os contornos pelo interior*;
- *separa elementos que estão próximos na imagem*;
- *elimina elementos menores que o ES*;
- *os elementos obtidos são semelhantes aos originais*;
- *a imagem obtida é mais regular, portanto, possui menos detalhes que a original.*

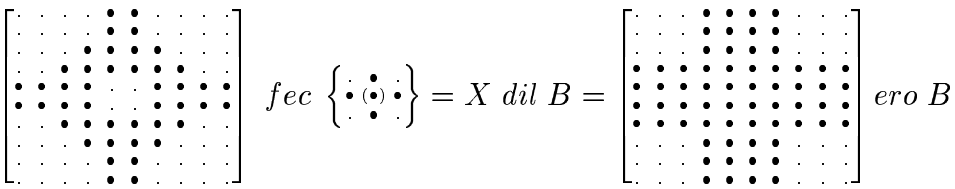
### Operação de Fechamento

A operação de fechamento[1] consiste no processo inverso ao da abertura, ou seja, primeiro é realizada a dilatação e depois a erosão.

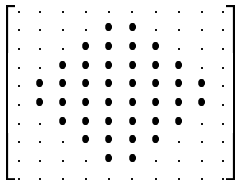
**Definição C.3.6** O fechamento de um conjunto $X$ pelo elemento estruturante $B$, denotado $X\ fec\ B$, é:

$$X\ fec\ B = (X\ dil\ B)\ ero\ B = (X \oplus \tilde{B}) \ominus \tilde{B}$$

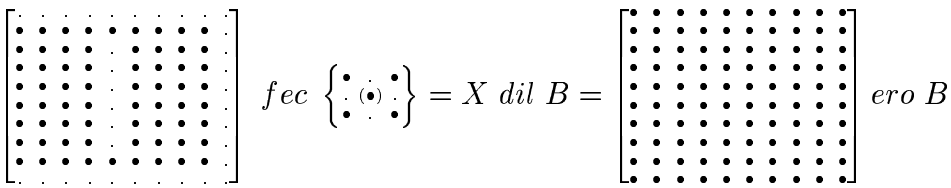
**Exemplo C.3.11** $X\ fec\ B$.

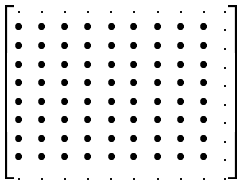


[1] A operação morfológica fechamento não está relacionada ao procedimento de fechamento de $PLEs$, definido na Seção 4.2.3, página 52.



**Exemplo C.3.12** $X\ fec\ B$.



De uma maneira geral, os seguintes efeitos podem ser obtidos com a operação fechamento:

- *nivela as fronteiras pelo exterior*;
- *preenche os buracos menores que o ES no interior da imagem*;
- *une elementos próximos na imagem*;
- *os elementos do resultado são semelhantes à imagem original*;
- *a imagem obtida é mais regular do que a original, logo, também possui menos detalhes.*

### C.3.4 Considerações Sobre os Operadores Apresentados

É importante ressaltar, que todas as operações apresentadas são referentes à morfologia binária, ou seja, aplicadas sobre imagens que possuem apenas dois tipos de pixels, *ativos* e *não-ativos*. Em [Facon, 1993] [Facon, 1996] [Weeks, 1996], são apresentados os conceitos de morfologia aplicada em imagens que possuem tons de cinza, porém, tal conteúdo está fora do escopo deste trabalho.

Algumas questões de eficiência também podem ser consideradas. Como descrito em [Facon, 1996], a utilização das operações de adição e subtração de Minkowski produz uma eficiência maior na execução dos algoritmos. Dentre os motivos que sustentam esse fato pode-se citar a implementação direta da interseção e união para as operações de erosão e dilatação, respectivamente, sem a necessidade de testá-las anteriormente e a rapidez com que as operações lógicas (AND, OR, etc.) são executadas.

Durante a apresentação das operações morfológicas, foi dito que as operações de erosão e dilatação deformam o tamanho da imagem original, e que as operações de abertura e fechamento tendem a manter o tamanho dos elementos da imagem. Porém, as operações de abertura e fechamento produzem um outro efeito que, muitas vezes, necessita ser eliminado. Elas modificam a forma da imagem original de modo que o resultado apresente forma semelhante à do elemento estruturante utilizado. Geralmente, essa deficiência é suprida considerando-se outros operadores morfológicos.