

---

Avaliação de métodos ótimos e  
subótimos de seleção de características  
de texturas em imagens

---

*Marco Aurélio Roncatti*

---



SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 17 de junho de 2008

Assinatura:

# Avaliação de métodos ótimos e subótimos de seleção de características de texturas em imagens

*Marco Aurélio Roncatti*

Orientador: *Prof. Dr. João do Espírito Santo Batista Neto*

Dissertação apresentada ao Instituto de Ciências Matemáticas  
e de Computação — ICMC-USP, como parte dos requisitos  
para obtenção do título de Mestre em Ciências — Ciências de  
Computação e Matemática Computacional.

**USP - São Carlos**  
**Junho/2008**



“Quando a última árvore for cortada; o último rio for envenenado; e o último peixe for pescado, nós vamos perceber que não podemos comer dinheiro.”

— provérbio indígena norte-americano



# Agradecimentos

---

Agradeço aos meus pais Humberto e Anna, pelo apoio, incentivo, amor e por me criarem e educarem.

Agradeço ao meu irmão Alessandro, pelos conselhos que me deu a respeito da vida e por me ensinar a usar o computador.

Agradeço à minha namorada Carol, pelo carinho, amor, por me esperar e pelas dicas quanto às normas.

Agradeço ao meu orientador João Batista, pela amizade, por me guiar pelo mestrado e por me deixar ajudá-lo a construir o quiosque.

Agradeço aos professores Alexandre Delbem, Castelo, Eduardo Raul, Franklina, Guilherme, Mário de Castro, Odemir e Sarita, pela amizade, por me ensinarem novas técnicas de programação, pelas dicas para o projeto de mestrado e pelos rodízios de pizzas.

Agradeço ao Marcos por me permitir trabalhar em seu projeto, pelos horários flexíveis e pelas caronas.

Agradeço aos meus companheiros de república, pelas conversas descontraídas, por dividirem o aluguel comigo e por me deixarem dormir.

Agradeço aos meus colegas de pós-graduação, em especial a André, Arnaldo, Cláudio, Dalcimar, Danilo, Davi, Jarbas, José Arnaldo, João Florindo, Márcio, Sérgio e Tiago Etiene, pela amizade, pelas dicas de computação e pelos passeios à cachoeira, corridas de *kart*, duelos de *paintball*.

Agradeço aos funcionários do ICMC, em especial a Ana Paula, Arly, Elizabeth, Laura, Paulinho e Sonia, pelo ótimo atendimento e profissionalismo.

Este trabalho teve o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).



# Sumário

---

---

<b>Sumário</b>	vii
<b>Lista de Siglas</b>	ix
<b>Resumo</b>	xi
<b>Abstract</b>	xiii
<b>1 Introdução</b>	1
1.1 Estrutura do Documento . . . . .	3
<b>2 Texturas</b>	5
2.1 Métodos Estatísticos . . . . .	5
2.1.1 Estatísticas de Primeira Ordem . . . . .	6
2.1.2 Matrizes de Co-ocorrência . . . . .	7
2.1.3 Função de Autocorrelação . . . . .	9
2.1.4 Matrizes de <i>Run Lengths</i> . . . . .	9
2.1.5 Espectros de Textura . . . . .	10
2.2 Métodos Geométricos . . . . .	10
2.2.1 Diagrama de Voronoi . . . . .	10
2.2.2 Métodos estruturais . . . . .	11
2.3 Métodos Baseados em Modelos . . . . .	11
2.3.1 Campos Aleatórios de Markov . . . . .	11
2.3.2 Fractais . . . . .	11
2.4 Métodos de Processamento de Sinais . . . . .	13
2.4.1 Transformada de Fourier . . . . .	13
2.4.2 Transformada de <i>Wavelet</i> . . . . .	13
2.4.3 Filtros de Gabor . . . . .	14
2.5 Considerações Finais . . . . .	19
<b>3 Seleção de Características</b>	21
3.1 Função Critério . . . . .	25
3.1.1 Distância de Bhattacharyya . . . . .	26
3.1.2 Distância de Jeffries-Matusita . . . . .	27
3.1.3 Classificador de Distância Mínima . . . . .	27

3.2	<i>Branch and Bound</i>	28
3.2.1	<i>Branch and Bound</i> básico	29
3.2.2	<i>Branch and Bound</i> ordenado	32
3.2.3	Cálculo Recursivo do Valor da Função Critério	34
3.2.4	Árvore de Busca Mínima	35
3.2.5	<i>Branch and Bound</i> Rápido	35
3.2.6	<i>Branch and Bound</i> com Previsão Parcial	38
3.2.7	Busca da Direita para a Esquerda	38
3.2.8	<i>Branch and Bound</i> Adaptativo	38
3.3	Busca Seqüencial	41
3.4	Redes Neurais Artificiais	43
3.4.1	<i>Multilayer Perceptron</i>	46
3.4.2	Medida de Saliência	47
3.5	Algoritmos Genéticos	48
3.5.1	Aplicação de Algoritmos Genéticos em Seleção de Características	50
3.6	Considerações Finais	51
<b>4</b>	<b>Nova Estratégia para o <i>Branch and Bound</i></b>	<b>53</b>
4.1	Aplicação da Estratégia Floresta	56
4.2	Comparação e Avaliação	59
<b>5</b>	<b>Experimentos e Resultados</b>	<b>61</b>
5.1	Classificação de Foto Aérea	62
5.2	Segmentação de Mosaicos	68
5.3	Segmentação de Imagens Médicas	74
<b>6</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>81</b>
<b>Referências</b>		<b>83</b>
<b>Apêndice A Características Utilizadas</b>		<b>89</b>

# **Lista de Siglas**

---

---

**BB:** *branch and bound*

**CBIR:** *content-based image retrieval*

**CDM:** classificador de distância mínima

**Caract.:** característica

**Exaus.:** busca exaustiva

**JM:** Jeffries-Matusita

**MRI:** *magnetic resonance imaging*

**PCA:** *principal component analysis*

**PTA( $l, r$ ):** *plus- $l$  take-away- $r$*

**SBFS:** *sequential floating backward selection*

**SBS:** *sequential backward selection*

**SFFS:** *sequential floating forward selection*

**SFS:** *sequential forward selection*

**WDBC:** *Wisconsin diagnostic breast cancer*



# Resumo

---

---

Características de texturas atuam como bons descritores de imagens e podem ser empregadas em diversos problemas, como classificação e segmentação. Porém, quando o número de características é muito elevado, o reconhecimento de padrões pode ser prejudicado. A seleção de características contribui para a solução desse problema, podendo ser empregada tanto para redução da dimensionalidade como também para descobrir quais as melhores características de texturas para o tipo de imagem analisada. O objetivo deste trabalho é avaliar métodos ótimos e subótimos de seleção de características em problemas que envolvem texturas de imagens. Os algoritmos de seleção avaliados foram o *branch and bound*, a busca exaustiva e o *sequential floating forward selection* (SFFS). As funções critério empregadas na seleção foram a distância de Jeffries-Matusita e a taxa de acerto do classificador de distância mínima (CDM). As características de texturas empregadas nos experimentos foram obtidas com estatísticas de primeira ordem, matrizes de co-ocorrência e filtros de Gabor. Os experimentos realizados foram a classificação de regiões de uma foto aérea de plantação de eucalipto, a segmentação não-supervisionada de mosaicos de texturas de Brodatz e a segmentação supervisionada de imagens médicas (MRI do cérebro). O *branch and bound* é um algoritmo ótimo e mais eficiente do que a busca exaustiva na maioria dos casos. Porém, continua sendo um algoritmo lento. Este trabalho apresenta uma nova estratégia para o *branch and bound*, nomeada floresta, que melhorou significativamente a eficiência do algoritmo. A avaliação dos métodos de seleção de características mostrou que os melhores subconjuntos foram aqueles obtidos com o uso da taxa de acerto do CDM. A busca exaustiva e o *branch and bound*, mesmo com a estratégia floresta, foram considerados inviáveis devido ao alto tempo de processamento nos casos em que o número de característica é muito grande. O SFFS apresentou os melhores resultados, pois, além de mais rápido, encontrou as soluções ótimas ou próximas das ótimas. Pôde-se concluir também que a precisão no reconhecimento de padrões aumenta com a redução do número de características e que os melhores subconjuntos freqüentemente são formados por características de texturas obtidas com técnicas diferentes.

**Palavras-chave:** Reconhecimento de padrões. Seleção de características. Texturas. *Branch and bound. Sequential floating forward selection.*



# Abstract

---

---

Texture features are efficient image descriptors and can be employed in a wide range of applications, such as classification and segmentation. However, when the number of features is considerably high, pattern recognition tasks may be compromised. Feature selection helps prevent this problem, as it can be used to reduce data dimensionality and reveal features which best characterise images under investigation. This work aims to evaluate optimal and suboptimal feature selection algorithms in the context of textural features extracted from images. Branch and bound, exhaustive search and sequential floating forward selection (SFFS) were the algorithms investigated. The criterion functions employed during selection were the Jeffries-Matusita (JM) distance and the minimum distance classifier (MDC) accuracy rate. Texture features were computed from first-order statistics, co-occurrence matrices and Gabor filters. Three different experiments have been conducted: classification of aerial picture of eucalyptus plantations, unsupervised segmentation of mosaics of Brodatz texture samples and supervised segmentation of MRI images of the brain. The branch and bound is an optimal algorithm and many times more efficient than exhaustive search. But is still time consuming. This work proposed a novel strategy for the branch and bound algorithm, named forest, which has considerably improved its performance. The evaluation of the feature selection methods has revealed that the best feature subsets were those computed by the MDC accuracy rate criterion function. Exhaustive search and branch and bound approaches have been considered unfeasible, due to their high processing times, especially for high dimensional data. This statement holds even for the branch and bound with the forest strategy. The SFFS approach yielded the best results. Not only was it faster, as it also was capable of finding the optimal or nearly optimal solutions. Finally, it has been observed that the precision of pattern recognition tasks increases as the number of features decreases and that the best feature subsets are those which possess features computed from distinct texture feature methods.

**Keywords:** Pattern recognition. Feature selection. Textures. Branch and bound. Sequential floating forward selection.



# Introdução

---

---

Muitas aplicações na área de análise de imagens fazem uso extenso de medidas obtidas de píxeis. No contexto de reconhecimento de padrões, tais medidas são denominadas características ou atributos. Normalmente são utilizadas características que correspondem a propriedades da cor, forma e textura da imagem (Gonzalez e Woods, 1992; Silva, 2006; Tuceryan e Jain, 1998; Zhang e Lu, 2004). A relevância de um conjunto de características é altamente vinculada à natureza das imagens e do problema abordado. Características de cor, por exemplo, podem resultar em uma alta taxa de acerto quando utilizadas para classificação de imagens naturais. Por outro lado, as mesmas características, se empregadas na identificação biométrica por imagens de íris, muito provavelmente não levarão à mesma precisão. Portanto, a escolha das características adequadas é fundamental para um reconhecimento de padrões eficiente.

Outro problema relacionado ao uso de características é a dimensionalidade. Um grande número de características freqüentemente prejudica o reconhecimento de padrões. Isto é, a taxa de acerto pode diminuir com o acréscimo de características em uma base de dados. Além disso, quanto maior o número de características, maior o custo computacional (memória e processamento). Logo, a redução da dimensionalidade é uma etapa importante do pré-processamento dos dados.

A redução do número de características pode ser conseguida com uma transformação das características originais. A abordagem mais conhecida para essa transformação é a análise de componentes principais (*principal component analysis*, PCA) (Fukunaga, 1990). Na literatura, essa abordagem é denominada extração de características. Nesta dissertação, o termo **extração de características** refere-se à obtenção de medidas a partir de texturas e **método de extração de características** refere-se ao algoritmo empregado na extração.

Outra abordagem para a redução da dimensionalidade é a **seleção de características**, que consiste na obtenção de um subconjunto com as características mais relevantes do conjunto original de acordo com certo critério. Uma **função critério** é então utilizada para medir a qualidade de um subconjunto. A função critério pode ser a taxa de acerto de um classificador ou uma medida estatística do grau de separação entre as classes da base de dados analisada. Sem perda de generalidade, assume-se que o objetivo da seleção é encontrar um subconjunto com o valor máximo para a função critério. Diversos **algoritmos de busca** podem ser utilizados para a seleção de características. Alguns algoritmos são **ótimos** (há a garantia de que a solução é a melhor possível), enquanto outros são **subótimos** (há apenas a intenção de encontrar a solução ótima ou próxima da ótima). O *branch and bound* é amplamente utilizado como algoritmo ótimo e existem diversas abordagens para algoritmos subótimos. Nessa dissertação, o termo **método de seleção de características** refere-se ao uso de um algoritmo de busca com uma função critério.

O objetivo deste trabalho é estudar diferentes métodos de seleção de características e avaliar como a seleção melhora o reconhecimento de padrões quando características de texturas de imagens são utilizadas. Apenas características de texturas foram utilizadas pois podem ser empregadas em diversos tipos de problemas e representam bem as particularidades das imagens. Estudos mostram que o uso de diferentes métodos para extração de características de texturas leva a uma maior precisão em reconhecimento de padrões se comparado com o uso de métodos isolados (Jain e Zongker, 1997). Em geral, as características de textura são contínuas, aproximam-se de uma distribuição normal, podem apresentar valores constantes (quando obtidas de regiões homogêneas) e são numerosas (o que justifica a redução de dimensionalidade). Assim, essas propriedades foram levadas em conta para a avaliação dos métodos de seleção.

Diferentes métodos de extração de características foram estudados. As características de texturas utilizadas nos experimentos foram obtidas com **matrizes de co-ocorrência** (Haralick et al., 1973), **filtros de Gabor** (Daugman e Downing, 1995) e **estatísticas de primeira ordem** (Materka e Strzelecki, 1998; Tuceryan e Jain, 1998). Alguns métodos de seleção de características foram estudados. Os algoritmos utilizados foram o ***branch and bound***, a **busca exaustiva** e o ***sequential forward floating selection*** (SFFS) (Pudil et al., 1994). O SFFS apresentou bons resultados em avaliações já realizadas e por isso é recomendado (Ferri et al., 1994; Kudo e Sklansky, 2000).

O *branch and bound*, além de ser avaliado, também foi empregado para verificar o quanto próximo o SFFS chegou da solução ótima. As diversas melhorias já apresentadas do *branch and bound* foram estudadas e uma nova estratégia foi proposta, chamada de **floresta**. O *branch and bound* utiliza uma árvore de busca para a seleção. A estratégia floresta consiste na utilização de mais de uma árvore busca, o que reduz o número de chamadas da função critério e, consequentemente, o tempo de execução.

Os efeitos da seleção de características foram avaliados em experimentos com

problemas reais (classificação de imagens aéreas e segmentação de imagens médicas) e problemas sintéticos (segmentação de mosaicos de texturas). Os algoritmos foram comparados de acordo com a velocidade de processamento e o valor obtido da função critério. A taxa de acerto obtida em testes com bases de dados diferentes das utilizadas na seleção também foi usada para comparação. Observou-se que a seleção de características contribuiu para aumentar a precisão em reconhecimento de padrões e que características obtidas por diferentes métodos de extração compõem os melhores subconjuntos. Isso mostra que os métodos utilizados para seleção foram satisfatórios.

## 1.1 Estrutura do Documento

No Capítulo 2 é feita uma revisão sobre métodos de extração de características de texturas. No Capítulo 3 é feita uma revisão sobre métodos de seleção de características, sendo que a Seção 3.1 trata das funções critério e as Seções 3.2–3.5 tratam dos algoritmos de busca. A estratégia floresta é apresentada no Capítulo 4, assim como o resultado de um experimento que mostra como a estratégia pode melhorar a eficiência do *branch and bound*. Esse experimento não emprega texturas de imagens. A descrição dos experimentos que envolvem texturas é feita no Capítulo 5. Os resultados das comparações dos diferentes métodos de seleção também são mostrados nesse capítulo. As avaliações finais acerca dos experimentos e das contribuições obtidas são apresentadas no Capítulo 6, juntamente com idéias para possíveis continuações desse trabalho.



# Texturas

---

---

Texturas são facilmente distinguíveis por observadores humanos e são importantes para o entendimento de imagens. Porém, são de difícil análise por sistemas computacionais e não existe uma definição formal para tal conceito. A definição proposta por Sklansky (1978) é: “Uma região de uma imagem possui uma textura constante se o conjunto de estatísticas locais ou outras propriedades locais da imagem são constantes, variam lentamente ou mantêm alguma periodicidade”. Vale destacar também que textura é uma propriedade de uma região da imagem. Portanto, a textura de um ponto não pode ser definida (Tuceryan e Jain, 1998).

Diversos trabalhos já foram realizados para estudos de texturas (Haralick et al., 1973; Galloway, 1975; He e Wang, 1990; Tuceryan e Jain, 1990; Lefebvre e Poulin, 2000; Comer e Delp, 2000; Zhou et al., 2001). Existem alguns métodos para obtenção de características intuitivas como granularidade, direcionalidade, aspereza e regularidade; e não intuitivas como energia, entropia e correlação. Essas características podem ser utilizadas para aplicações como segmentação, classificação, recuperação de imagens e síntese de texturas.

Na revisão realizada por Tuceryan e Jain (1998), alguns dos métodos mais conhecidos de análise de texturas são divididos de acordo com quatro abordagens: estatística, geométrica, baseada em modelos e baseada em processamento de sinais. Os métodos apresentados a seguir são organizados com essa mesma taxonomia.

## 2.1 Métodos Estatísticos

Medições estatísticas têm sido usadas desde os primeiros trabalhos com textura (Julesz, 1962). Podem ser de **primeira ordem**, em que são avaliadas as probabilidades de se

encontrar píxeis das diversas tonalidades possíveis na imagem; ou de **segunda ordem**, em que o posicionamento dos píxeis também é levado em conta.

### 2.1.1 Estatísticas de Primeira Ordem

O histograma da região que se deseja analisar é utilizado (Tuceryan e Jain, 1998; Materka e Strzelecki, 1998). A vantagem é o baixo custo computacional. Porém, não é raro que texturas diferentes apresentem histogramas muito parecidos.

Seja a matriz  $\mathbf{I}$  a imagem analisada. O valor do elemento  $I(m, n)$  corresponde ao valor do píxel na  $m$ -ésima linha e  $n$ -ésima coluna da imagem, sendo que  $m = 0, 1, \dots, M-1$  e  $n = 0, 1, \dots, N-1$ . Dessa maneira, se  $G$  é o número de cores da imagem,  $I(m, n) = 0, 1, \dots, G-1$ . O histograma  $h(\cdot)$  da imagem  $\mathbf{I}$  é definido por

$$h(i) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i, I(m, n)), \quad i = 0, 1, \dots, G-1, \quad (2.1)$$

$$\delta(i, j) = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j. \end{cases} \quad (2.2)$$

A função  $\delta(i, j)$  é chamada delta de Kronecker. O valor de  $h(i)$  corresponde ao número de ocorrências da cor  $i$  na imagem. A função massa de probabilidade  $p(i)$ , que representa a probabilidade de ocorrência da cor  $i$ , é obtida dividindo-se  $h(i)$  pelo número total de píxeis da imagem:

$$p(i) = h(i) / (N \cdot M). \quad (2.3)$$

Diversas medidas podem ser extraídas do histograma. Algumas das medidas mais utilizadas são apresentadas nas Equações 2.4–2.9.

$$\text{Média:} \quad \mu = \sum_{i=0}^{C-1} i \cdot p(i) \quad (2.4)$$

$$\text{Variância:} \quad \sigma^2 = \sum_{i=0}^{C-1} (i - \mu)^2 \cdot p(i) \quad (2.5)$$

$$\text{Obliquidade:} \quad \gamma_1 = \frac{1}{\sigma^3} \cdot \sum_{i=0}^{C-1} (i - \mu)^3 \cdot p(i) \quad (2.6)$$

$$\text{Curtose:} \quad \gamma_2 = \left( \frac{1}{\sigma^4} \cdot \sum_{i=0}^{C-1} (i - \mu)^4 \cdot p(i) \right) - 3 \quad (2.7)$$

Energia: 
$$E = \sum_{i=0}^{C-1} (p(i))^2 \quad (2.8)$$

Entropia: 
$$H = - \sum_{i=0}^{C-1} p(i) \cdot \log_2(p(i)) \quad (2.9)$$

A **média** e a **variância** são conceitos muito comuns em estatística e representam, respectivamente, a cor média e a dispersão em relação à média no histograma. A **obliquidade** corresponde ao grau de simetria. Se  $\gamma_1 = 0$ , o histograma é simétrico em relação à média. Caso contrário, a distribuição concentra-se à esquerda ou à direita da média. A **curtose** relaciona-se com o “achatamento” do histograma. Se  $\gamma_2 = 0$ , o achatamento é o mesmo de uma distribuição normal. Se  $\gamma_2 > 0$ , o histograma é mais alto que uma normal. Se  $\gamma_2 < 0$ , é mais baixo que uma normal. A **energia** mede a presença de valores altos (em relação aos demais valores) no histograma e a **entropia** mede a uniformidade do histograma.

### 2.1.2 Matrizes de Co-ocorrência

Matrizes de co-ocorrência (Haralick et al., 1973) estão entre os métodos mais utilizados para extração de características de texturas. As matrizes são construídas pela comparação da cor de píxeis situados a uma determinada posição uns dos outros. Diferentes matrizes de co-ocorrências são criadas para a mesma imagem. Algumas medidas são então obtidas dessas matrizes.

Sendo  $G$  o número de cores da imagem, cada matriz de co-ocorrência tem tamanho  $G \times G$ . A comparação dos píxeis é feita de acordo com um deslocamento horizontal  $d_x$  e vertical  $d_y$ . Seja  $\mathbf{C}_{d_x d_y}$  uma matriz de co-ocorrência para os deslocamentos  $d_x$  e  $d_y$ . O elemento  $C_{d_x d_y}(i, j)$  corresponde ao número de ocorrências de píxeis da cor  $j$  situados a um deslocamento horizontal  $d_x$  e vertical  $d_y$  de píxeis da cor  $i$ . Ou seja, a ocorrência é considerada apenas quando  $I(m, n) = i$  e  $I(m + d_y, n + d_x) = j$ . Utilizando-se notação semelhante à apresentada na seção 2.1.1, a geração de matrizes de co-ocorrências é definida por

$$C_{d_x d_y}(i, j) = \sum_{m=-\min(0, d_y)}^{M-1-\max(0, d_y)} \sum_{n=-\min(0, d_x)}^{N-1-\max(0, d_x)} \delta(i, I(m, n)) \cdot \delta(j, I(m + d_y, n + d_x)). \quad (2.10)$$

As funções  $\min(\cdot)$  e  $\max(\cdot)$ , presentes nos limites dos somatórios, são necessárias para garantir que os píxeis  $I(m, n)$  e  $I(m + d_y, n + d_x)$  pertençam à imagem. A Figura 2.1 mostra uma imagem e três exemplos de matrizes de co-ocorrência geradas a partir dessa imagem. As características extraídas de duas matrizes com deslocamentos opostos são iguais ou muito próximas. Por isso, costuma-se realizar a soma dessas matrizes para

$$\mathbf{I} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 3 & 3 & 2 & 2 & 1 \\ 0 & 3 & 3 & 2 & 2 & 1 \\ 0 & 3 & 3 & 2 & 2 & 1 \end{bmatrix} \quad \mathbf{C}_{10} = \begin{bmatrix} 2 & 2 & 0 & 3 \\ 0 & 6 & 0 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 3 & 3 \end{bmatrix}$$

$$\mathbf{C}_{-11} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 3 & 4 & 2 \\ 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 2 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 3 & 0 & 0 & 2 \\ 0 & 3 & 4 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Figura 2.1: Exemplos de matrizes de co-ocorrência. A imagem  $\mathbf{I}$  está representada como matriz e possui quatro cores.

se obter uma matriz simétrica ( $\mathbf{C}_{d_x d_y} + \mathbf{C}_{-d_x -d_y}$ ). Quanto maior o número de cores, mais esparsa é a matriz de co-ocorrência. Portanto, reduzir do número de cores é uma boa maneira melhorar o desempenho do algoritmo prejudicando pouco a qualidade das características. Antes da aplicação das fórmulas para extração das características, as matrizes devem ser normalizadas. Com isso, obtém-se a probabilidade de ocorrência de pares de pixels para cada posicionamento. A matriz normalizada  $\mathbf{R}_{d_x d_y}$  é obtida por

$$\mathbf{R}_{d_x d_y}(i, j) = \frac{\mathbf{C}_{d_x d_y}(i, j)}{\sum_{m=0}^{G-1} \sum_{n=0}^{G-1} \mathbf{C}_{d_x d_y}(m, n)}. \quad (2.11)$$

A maneira como os valores estão distribuídos na matriz reflete características da textura. Por exemplo, texturas com pouco contraste resultam em valores mais altos próximos ao eixo da matriz. Haralick et al. (1973) apresentam 14 fórmulas para extração de características. Porém, apenas seis são consideradas relevantes (Cossu, 1998 apud Baraldi e Parmiggiani, 1995). Alguns autores possuem interpretações diferentes para algumas fórmulas (Haralick et al., 1973; Tuceryan e Jain, 1998; Baraldi e Parmiggiani, 1995). As fórmulas a seguir foram sugeridas por Baraldi e Parmiggiani (1995) e são válidas para matrizes de co-ocorrência simétricas.

$$\text{Energia:} \quad F_1 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (\mathbf{R}_{d_x d_y}(i, j))^2, \quad (2.12)$$

$$\text{Contraste:} \quad F_2 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i - j)^2 \cdot \mathbf{R}_{d_x d_y}(i, j), \quad (2.13)$$

Correlação: 
$$F_3 = \frac{\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i - \mu) \cdot (j - \mu) \cdot R_{d_x d_y}(i, j)}{\sigma^2}, \quad (2.14)$$

Variância: 
$$F_4 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i - \mu)^2 \cdot R_{d_x d_y}(i, j), \quad (2.15)$$

Momento da Diferença Inversa: 
$$F_5 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{R_{d_x d_y}(i, j)}{1 + (i - j)^2}, \quad (2.16)$$

Entropia: 
$$F_6 = - \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} R_{d_x d_y}(i, j) \cdot \log_2 (R_{d_x d_y}(i, j)), \quad (2.17)$$

sendo que

$$\mu = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} i \cdot R_{d_x d_y}(i, j), \quad \sigma^2 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i - \mu)^2 \cdot R_{d_x d_y}(i, j). \quad (2.18)$$

Características invariantes em relação à orientação das texturas também podem ser obtidas das matrizes de co-ocorrência (Haralick et al., 1973; Santos, 2007). Para isso, são aplicadas operações estatísticas em características obtidas de matrizes de deslocamentos com distâncias próximas e orientações diferentes. Por exemplo, pode-se calcular a média e o desvio padrão da energia das matrizes  $\mathbf{R}_{1\ 0}$ ,  $\mathbf{R}_{1\ 1}$ ,  $\mathbf{R}_{0\ 1}$  e  $\mathbf{R}_{-1\ 1}$ .

### 2.1.3 Função de Autocorrelação

A função de autocorrelação corresponde ao valor da correlação entre a imagem e uma cópia da imagem deslocada horizontal e verticalmente. Um comportamento característico é obtido em texturas que possuem alguma regularidade (Tuceryan e Jain, 1998). Ou seja, a função apresenta máximos locais correspondentes à disposição das primitivas que formam a textura. A obtenção de resultado semelhante também é possível a partir do espectro de potência da transformada de Fourier.

### 2.1.4 Matrizes de *Run Lengths*

Uma matriz de *run lengths* (Galloway, 1975) é construída pela contagem de seqüências retilíneas de píxeis da região analisada que possuem a mesma tonalidade, sendo que cada linha da matriz corresponde a uma tonalidade e cada coluna, a um comprimento. A direção (normalmente  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  e  $135^\circ$ ) é fixa para cada matriz. Como os caminhos de comprimento mais longo são menos freqüentes, as colunas podem ser agrupadas em

intervalos logaritmos (1, 2–3, 4–7, 8–15, ...) (Albregtsen et al., 2000). Algumas das características que podem ser obtidas a partir das matrizes são: influência dos caminhos curtos, influência dos caminhos longos, grau de não-uniformidade de tons e grau de não-uniformidade de comprimento de caminho.

### 2.1.5 Espectros de Textura

He e Wang (1990) propõem o conceito de unidade de textura, formada por um píxel central e os oito píxeis vizinhos. Uma unidade é rotulada de acordo com a comparação entre o valor da tonalidade do píxel central e de seus vizinhos, sendo que três valores são possíveis para a comparação: maior, menor ou igual. Essas comparações são usadas para a geração de um número entre zero e 6561 (pois  $8^3$  arranjos são possíveis). O espectro de textura é construído pela avaliação da freqüência de cada número das unidades de textura de uma região da imagem. Alguns valores que podem ser obtidos a partir do espectro são propostos por He e Wang (1991). Uma versão binária para a geração do número das unidades de textura é apresentada por Ojala et al. (1996).

## 2.2 Métodos Geométricos

Em métodos geométricos, as texturas são caracterizadas por uma composição de primitivas, também chamadas de *textel* ou *texton*. Nessa classe de métodos, medidas podem ser extraídas tanto das características das primitivas como da forma como estão posicionadas na imagem compondo a textura.

### 2.2.1 Diagrama de Voronoi

Tuceryan e Jain (1990) sugerem o uso do diagrama de Voronoi para segmentar textura. É possível compreender a formação dos polígonos que integram o diagrama de Voronoi considerando-se que cada primitiva de textura é constituída por um ponto isolado. Elaborar um diagrama de Voronoi consiste em particionar o plano com um polígono para cada ponto. A partição surge das regiões resultantes ao se traçar uma reta divisória entre o ponto dado e cada um de seus vizinhos. A intersecção dos semiplanos resultantes que passam pelo ponto é a área interna do polígono de Voronoi. Quando todos os pontos da imagem foram utilizados, o diagrama está completo. A triangulação de Delaunay pode ser obtida ligando-se cada par de pontos de partições vizinhas. O momento de área desses polígonos pode ser usado como característica da textura, refletindo tanto a forma quanto a distribuição espacial das primitivas.

### **2.2.2 Métodos estruturais**

Em geral, essa classe de algoritmos só funciona para texturas bastante regulares. Ela envolve duas etapas: extração dos elementos de textura e inferência da regra de posicionamento. Usualmente, os elementos de textura são regiões de tonalidade homogênea. Uma das propostas de uso desse método é feita por Lefebvre e Poulin (2000), em que é realizado um pré-processamento por meio da detecção de bordas, limiarização e operações morfológicas para geração de uma máscara. Um identificador de freqüência é aplicado sobre a máscara, obtendo-se o tamanho e orientação das primitivas da textura.

## **2.3 Métodos Baseados em Modelos**

Diferentemente dos métodos estruturais e semelhantemente aos métodos estatísticos, os métodos descritos nesta seção tratam a textura como sendo regida por um padrão de distribuição dos píxeis e não de primitivas maiores. Realmente, existem imagens, como um campo gramado visto de longe ou o interior de uma nuvem, nas quais os próprios humanos são incapazes de identificar elementos de contorno definido que se repitam por toda uma região. Ainda assim elas apresentam textura e são mais bem trabalhadas por modelos que estudam a forma como os píxeis são distribuídos, incluindo características contextuais do espaço e estatísticas. Campos aleatórios de Markov e fractais, apresentados a seguir, são dois exemplos de modelos.

### **2.3.1 Campos Aleatórios de Markov**

O modelo por campos aleatórios de Markov (Comer e Delp, 2000) tem sido aplicado em síntese, classificação, segmentação, restauração e compressão de imagens (Tuceryan e Jain, 1998). Ele assume que a intensidade de cada píxel na imagem depende probabilisticamente das intensidades dos píxeis vizinhos. Cada píxel é visto como uma variável aleatória, considerando, assim, a imagem como um campo aleatório. Ou seja, um grafo com os vértices sendo os píxeis e as arestas sendo a ligação entre os píxeis vizinhos. Probabilidades condicionais são definidas usando-se cliques que podem ser simplesmente um píxel, um par de píxeis horizontal, vertical ou triplas de píxeis que estejam dentro da vizinhança do píxel em análise. Os campos de Markov funcionam bem para microtexturas, mas apresentam problemas em texturas regulares ou muito heterogêneas. Nesses casos, uma abordagem multi-escala pode aperfeiçoar o método (Gerhardinger, 2006; Comer e Delp, 2000).

### **2.3.2 Fractais**

Fractais são representações gráficas de fenômenos caóticos (Mandelbrot, 1983). Cada parte de um fractal assemelha-se a uma outra parte maior. Em outras palavras, possuem auto-

semelhança em escala. Algumas formas na natureza apresentam propriedades parecidas às dos fractais. O que explica, em alguns casos, a utilização de métodos da geometria fractal na extração de características de imagens. Como fractais são objetos matemáticos, os métodos aplicados em imagens são adaptações dos métodos originais. Duas características podem ser extraídas de imagens: dimensão fractal e lacunaridade.

Enquanto na geometria euclidiana a dimensão é um valor inteiro, na geometria fractal esse valor é fracionário. Um dos métodos mais utilizados para a extração da **dimensão fractal** de imagens é o **box-counting** (Sarkar e Chaudhuri, 1992) devido a simplicidade e fácil implementação. Uma malha de quadrados é sobreposta à imagem. O número de quadrados pelos quais a forma analisada passa deve ser contada. A equação a seguir define a dimensão fractal  $D$ :

$$D = -\lim_{r \rightarrow 0} \frac{\ln(N_r(A))}{\ln(r)} \quad (2.19)$$

em que  $r$  é o lado dos quadrados,  $A$  é a forma analisada e  $N_r(A)$  é o número de quadrados contados. Como a aplicação em imagens é um caso discreto, não é possível encontrar o limite da equação. Para solução do problema, um gráfico log-log deve ser traçado com os valores de  $\ln(r) \times \ln(N_r(A))$ . O valor da dimensão fractal é determinado por  $D = -\alpha$ , sendo que  $\alpha$  é o coeficiente angular da reta que melhor aproxima-se dos pontos do gráfico.

Esse processo do box-counting pode ser aplicado apenas em imagens binárias. Para imagens em tons de cinza, deve-se utilizar o box-counting 3D. O nível de cinza representa a terceira dimensão. A imagem é então analisada como um sólido. Em vez de quadrados, uma malha de cubos de lado  $r$  deve ser utilizada.

A **lacunaridade** é complementar à dimensão fractal e mede o quanto um fractal ocupa o espaço. O algoritmo **gliding-box** (Plotnick et al., 1996), similar ao box-counting, pode ser utilizado para o cálculo dessa característica. Um quadrado de lado  $r$  deve ser posicionado em cada linha e coluna da imagem. O número de pontos da imagem presentes dentro do quadrado deve ser contado. É gerada assim uma distribuição de freqüência da massa  $n(s, r)$ , em que  $s$  é o número de pontos internos ao quadrado. Dividindo-se essa função pelo número total de quadrados de tamanho  $r$  utilizados, obtém-se a distribuição de probabilidade  $Q(s, r)$ . Isso quer dizer que  $Q(s, r)$  é a probabilidade de um quadrado de lado  $r$  conter  $s$  pontos da imagem. Aplicando-se as Equações 2.20–2.22, o valor da lacunaridade  $\Lambda(r)$  é determinado.

$$Z_1(r) = \sum_{s=0}^{r^2} sQ(s, r) \quad (2.20)$$

$$Z_2(r) = \sum_{s=0}^{r^2} s^2 Q(s, r) \quad (2.21)$$

$$\Lambda(r) = \frac{Z_2(r)}{Z_1(r)^2} \quad (2.22)$$

Assim como é feito com o box-counting, para aplicação em imagens em tons de cinza deve-se utilizar a versão tridimensional do gliding-box, que consiste na utilização de cubos no lugar de quadrados.

## 2.4 Métodos de Processamento de Sinais

Os métodos apresentados nesta seção detectam freqüências, que podem ser entendidas como a medida da taxa de repetição de determinado padrão na imagem.

### 2.4.1 Transformada de Fourier

A transformada de Fourier discreta (Brigham, 1974), quando aplicada à função de um sinal, retorna valores que expressam a função original em termos de funções de base senoidal. Cada função retornada pela transformada corresponde a uma freqüência. A soma dessas funções é igual à função original. Zhou et al. (2001) propõem uma das formas de se aplicar a transformada para extração de característica de textura. Para cada píxel, consideram-se os oito píxeis vizinhos mais próximos como o vetor de entrada da transformada. Com esse método é possível obter informações locais sobre a variação dos níveis cinza. Intuitivamente, essas informações estão relacionadas à rugosidade da imagem. Outra forma de se aplicar a transformada é por meio da utilização de sua versão em duas dimensões, para identificar as freqüências em todas as direções de cada pequena região quadrada da imagem.

### 2.4.2 Transformada de *Wavelet*

A transformada de *wavelet* (Walker, 1999) consiste na representação de um sinal em termos de um outro sinal, chamado *wavelet* mãe. A parte significativa de uma *wavelet* mãe possui comprimento finito e decaimento nas extremidades. O termo *wavelet* vem do francês *ondelette* e significa onda pequena. Aqui será abordada apenas a transformada de *wavelet* discreta.

A saída de uma transformada unidimensional de *wavelet* é dividida em duas partes de igual comprimento: tendência e detalhe. O comprimento total da saída é igual ao do

sinal original. Cada valor da **tendência** é obtido pelo produto escalar entre o vetor do sinal e um vetor chamado **base escala**. Para os valores do **detalhe**, o produto escalar é feito com um vetor chamado **base wavelet**. Esse processo pode ser repetido na tendência, gerando mais um nível de transformação, e repetido quantas vezes for desejado para cada nova tendência. Outra abordagem, chamada *wavelet packets*, consiste na repetição da transformação tanto na tendência como no detalhe. Existem diversas bases, cada uma é apropriada para sinais de tipos diferentes.

As bases escala e *wavelet* são ortogonais. Com isso, os dados contidos no sinal original são preservados após a transformação. Outra importante propriedade da transformada de *wavelet* é a conservação de energia. A energia é calculada pela soma dos quadrados dos elementos de um vetor. Em outras palavras, a energia do sinal original é o mesmo do sinal resultante da transformada de *wavelet*. O valor da energia pode ser utilizado, por exemplo, na compactação de dados: é definida inicialmente a quantidade de energia do sinal original que deve ser preservada no sinal compactado. A transformada de *wavelet* concentra a energia na tendência. Portanto, são preservados no sinal compactado apenas os elementos do vetor da transformada necessários para se atingir a energia definida.

A transformada *wavelet* bidimensional é utilizada em imagens. Essa variação consiste na aplicação da transformada unidimensional em cada linha e depois em cada coluna da imagem, ou o contrário, primeiro em cada coluna e depois em cada linha, gerando quatro regiões: a tendência, o detalhe horizontal, o detalhe vertical e o detalhe diagonal. Para os outros níveis, o mesmo processo é aplicado na tendência do nível anterior. A Figura 2.2 mostra um exemplo da aplicação da transformada de Coif6 para 1, 2 e 3 níveis.

As diferentes regiões da transformada *wavelet* bidimensional correspondem a determinadas propriedades da textura. Por exemplo, a região de detalhe horizontal apresenta valores maiores em pontos onde há predomínio de freqüências altas horizontalmente. E a faixa de freqüência é diferente em cada nível da transformada. Portanto, as características de textura podem ser obtidas pelo cálculo da energia em partes da imagem. Realizando-se rotações na imagem e utilizando-se *wavelet packets*, um maior número de características pode ser obtido.

### 2.4.3 Filtros de Gabor

A função de Gabor (Gabor, 1946) é um sinal de uma dimensão formado pela adição de uma gaussiana a um sinal harmônico de determinada freqüência. Um sinal qualquer pode ser decomposto como a combinação de funções de Gabor de diferentes freqüências. Dessa maneira, é possível determinar a intensidade de cada freqüência em cada instante do sinal. A função de Gabor de duas dimensões é uma modificação da função unidimensional e

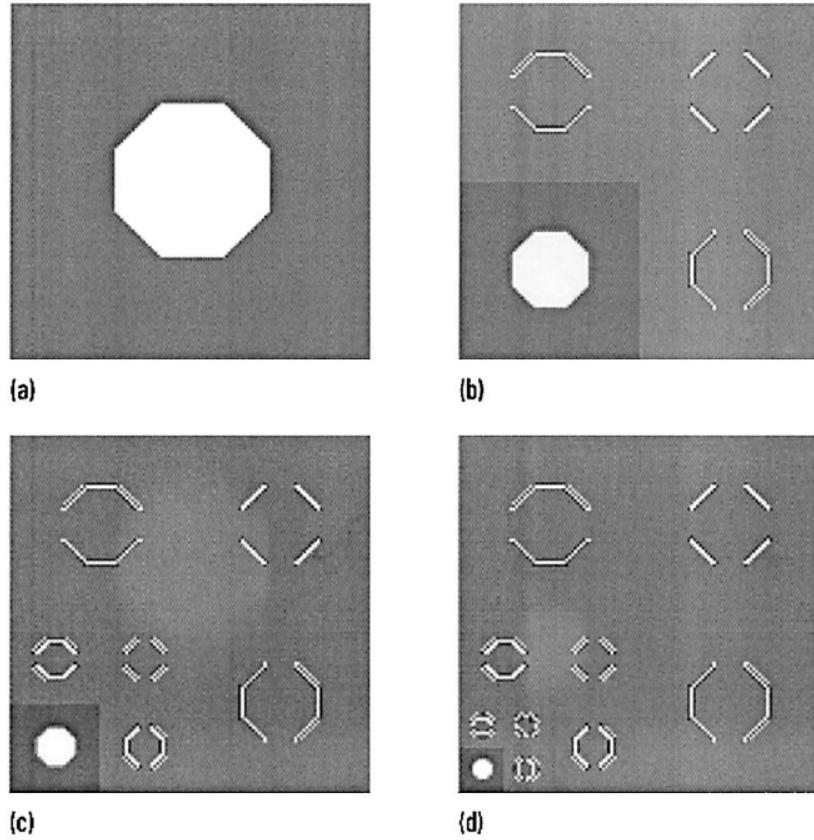


Figura 2.2: (a) Imagem original. (b) Transformada de Coif6 para 1 nível. (c) Transformada de Coif6 para 2 níveis. (d) Transformada de Coif6 para 3 níveis. (Walker, 1999)

assemelha-se a padrões biológicos existentes na visão de mamíferos (Daugman, 1980 apud Daugman e Downing, 1995). A função de duas dimensões pode ser utilizada na extração de características de imagens, permitindo a identificação da intensidade de freqüências em diferentes orientações. Filtros de Gabor ou *wavelets* de Gabor são as funções de Gabor quando aplicadas em extração de características. Aqui, o termo “*wavelet*” não tem relação com a divisão de sinais em tendência e detalhe como é feito com *wavelets* ortogonais.

Na retina de mamíferos, impulsos nervosos emitidos por cones e bastonetes são combinados por neurônios formando campos receptivos. Campos receptivos com características diferentes são espalhados pela retina de maneira ordenada. Resumidamente, os campos receptivos emitem impulsos nervosos quando a região correspondente da retina recebe luz no centro e não recebe luz na borda ou o contrário, recebe luz na borda e não recebe luz no centro. Os impulsos de campos receptivos alinhados também são combinados, formando outro padrão de reconhecimento na retina. Neste caso, o padrão é muito semelhante ao formado por funções de Gabor (Daugman, 1988).

Os filtros de Gabor utilizados para decompor uma imagem são gerados a partir

da *wavelet* de Gabor mãe

$$g(x, y) = \left( \frac{1}{2\pi\sigma_x\sigma_y} \right) \cdot \exp \left( -\frac{1}{2} \cdot \left( \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) + 2\pi\iota Wx \right), \quad (2.23)$$

sendo que

$$\iota^2 = -1,$$

$\sigma_x$  é o desvio padrão ao longo do eixo  $x$ ,

$\sigma_y$  é o desvio padrão ao longo do eixo  $y$ ,

$W$  é a freqüência central.

A freqüência do sinal harmônico (freqüência central) é a mesma de  $\cos(2\pi Wx)$ . Em uma imagem definida como uma função discreta,  $W = 0,5$  é a maior freqüência que pode ser representada no sentido do eixo  $x$ . Portanto, na maioria dos casos,  $W$  deve ser menor do que 0,5.

Um filtro, quando aplicado em uma imagem, extraia a intensidade de determinadas freqüências em cada píxel. Diversos filtros podem ser gerados pela rotação e mudança dos parâmetros de  $g(x, y)$ . Assim, um conjunto de filtros distintos pode ser utilizado para extração de diferentes características da imagem. Porém, os filtros de um conjunto não são ortogonais. Com isso, há redundância entre os filtros (informações da imagem são extraídas repetidamente por mais de um filtro) e a abrangência não é completa (pouca ou nenhuma informação é extraída para algumas freqüências e orientações).

Manjunath e Ma (1996) sugerem uma estratégia para geração do conjunto de filtros de maneira eficiente, diminuindo a redundância e aumentando a abrangência. Essa estratégia é explicada a seguir\*. Cada filtro de Gabor é definido por

$$g_{mn}(x, y) = a^{-m}g(x', y'), \\ m = 0, 1, \dots, S-1, \quad n = 0, 1, \dots, K-1,$$

sendo que

$S$  é número de freqüências centrais,

$K$  é número de orientações,

$a^{-m}$  é um fator de escala para garantir que a energia de  $g_{mn}(x, y)$  seja independente de  $m$  (a definição de  $a$  será apresentada mais a frente),

$x'$  e  $y'$  representam as coordenadas  $x$  e  $y$  rotacionadas.

---

\*Algumas fórmulas presentes no artigo em que essa estratégia foi apresentada contêm erros e não coincidem com as fórmulas presentes em outro artigo (Ferrari et al., 2004).

A rotação é realizada por

$$x' = a^{-m} (x \cos(\theta) + y \sin(\theta)), \quad y' = a^{-m} (-x \sin(\theta) + y \cos(\theta)), \\ \theta = \frac{n\pi}{K}.$$

Seja  $G_{mn}(u, v)$  a transformada de Fourier de  $g_{mn}(x, y)$  e  $\text{mag}(G_{mn}(u, v))$  a magnitude da transformada.  $\text{mag}(G_{mn}(u, v))$  é uma função gaussiana deslocada da origem de acordo com a freqüência central e orientação do filtro  $g_{mn}(x, y)$ . As variâncias dessa função gaussiana são  $\sigma_u^2$  e  $\sigma_v^2$ . Quanto maior o valor de  $\sigma_u^2$ , maior a extensão da função no sentido das freqüências (distância da origem). O valor de  $\sigma_v^2$  corresponde à extensão no sentido perpendicular ao sentido das freqüências. As variâncias  $\sigma_u^2$  e  $\sigma_v^2$  são inversamente proporcionais a  $\sigma_x^2$  e  $\sigma_y^2$ :

$$\sigma_u = \frac{1}{2\pi\sigma_x}, \quad \sigma_v = \frac{1}{2\pi\sigma_y}.$$

A maneira como os filtros se sobrepõem no domínio da freqüência é determinada pelo posicionamento da função gaussiana e pelos valores de  $\sigma_u^2$  e  $\sigma_v^2$ . A estratégia apresentada assegura que as elipses formadas pela intersecção de  $\text{mag}(G_{mn}(u, v))$  com o plano  $P_{mn}(u, v) = \max(\{\text{mag}(G_{mn}(i, j)) | (i, j) \in \mathbb{R}^2\})/2$  “tocam-se” quando os filtros são consecutivos. Para tal, são utilizadas as fórmulas

$$a = \left( \frac{U_h}{U_l} \right)^{\frac{1}{s-1}}, \quad \sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln(2)}}, \\ \sigma_v = \tan\left(\frac{\pi}{2K}\right) \left( U_h - 2\ln(2) \left( \frac{\sigma_u^2}{U_h} \right) \right) \left( 2\ln(2) - \frac{(2\ln(2))^2\sigma_u^2}{U_h^2} \right)^{-\frac{1}{2}},$$

sendo que  $U_l$  e  $U_h$  são as freqüências centrais inferior e superior, respectivamente. A freqüência central da Equação (2.23) deve ser  $W = U_h$ . A Figura 2.3 mostra alguns exemplos de conjuntos de filtros no domínio da freqüência. Observa-se na figura como as funções gaussianas distribuem-se pela área da transformada de Fourier e como as elipses correspondentes tocaram-se. Alguns filtros no domínio do espaço podem ser vistos na Figura 2.4. Nota-se que a parte real de um filtro tem simetria par, como a função cosseno em relação a origem, e a parte imaginária tem simetria ímpar, como a função seno.

Segundo Daugman (2003), é comum ocorrer na literatura o uso incorreto de termos sobre a utilização de filtros de Gabor. A **expansão de coeficientes** é a decomposição da imagem em filtros de Gabor de maneira que a imagem possa ser reconstruída pela combinação linear dos filtros. É correto chamar essa utilização de transformada de Gabor, pois existe operação inversa. Essa transformação pode ser feita utilizando-se redes

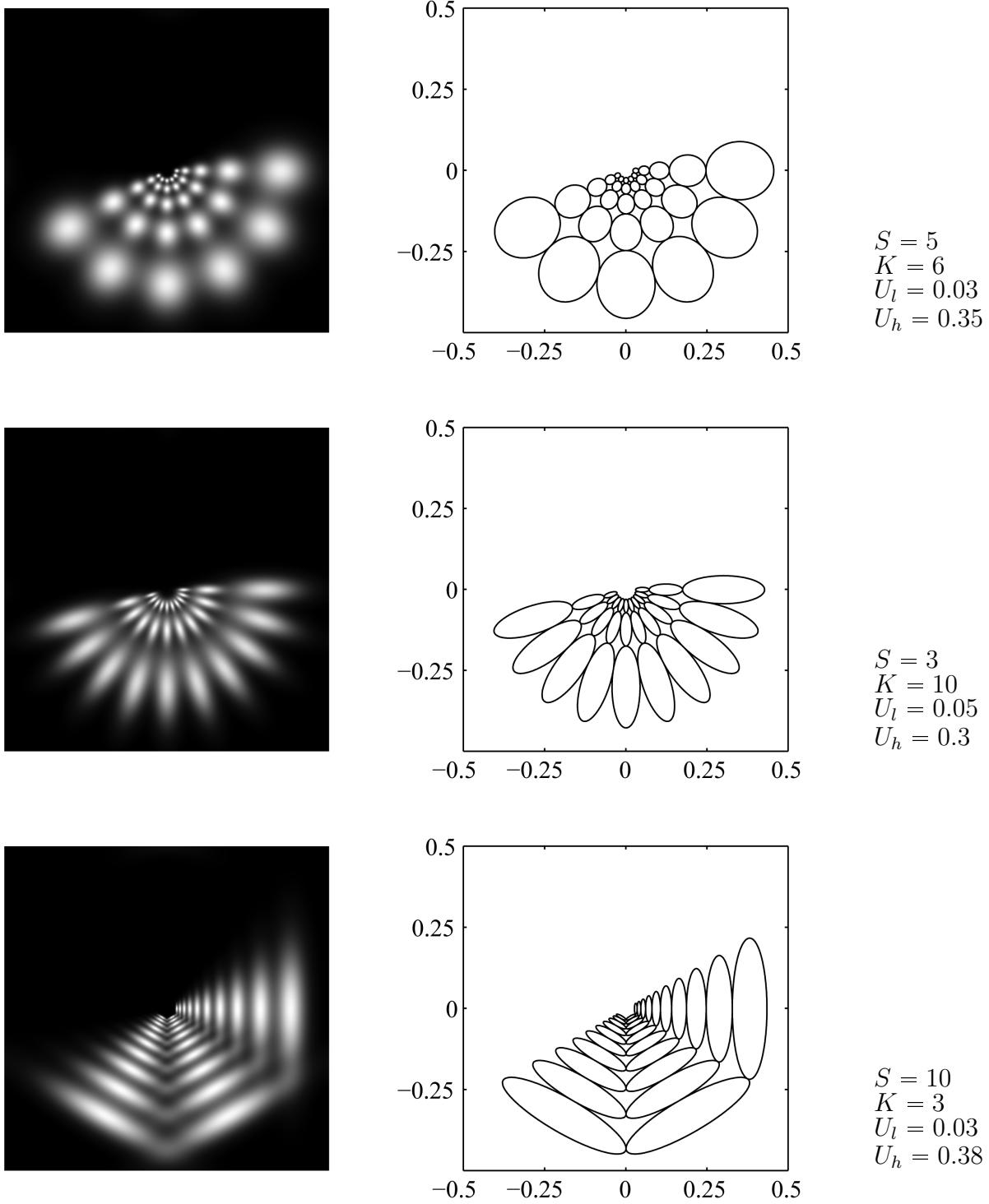


Figura 2.3: Conjuntos de filtros de Gabor no domínio da freqüência. Os eixos horizontais correspondem a  $u$  e os verticais, a  $v$ . A coluna da esquerda da tabela mostra os resultados de  $\sum_{m=0}^{S-1} \sum_{n=0}^{K-1} (G'_{mn}(\cdot) / \max(G'_{mn}(\cdot)))$ , sendo que  $G'_{mn}(\cdot) = \text{mag}(G_{mn}(u, v))^2$ . A coluna central mostra as elipses geradas pela intersecção de  $\text{mag}(G_{mn}(u, v))$  com o plano  $P_{mn}(u, v)$  para cada filtro. A coluna da direita mostra os parâmetros utilizados para gerar cada conjunto de filtros.

neurais (Daugman, 1988). Um obstáculo para a expansão de coeficientes é o alto custo computacional.

A **projeção de coeficientes** diz respeito a convolução entre os filtros e a imagem analisada. A imagem filtrada é formada por números complexos. Portanto, deve ser calculada a magnitude do resultado da convolução (Clark et al., 1987). Com isso, obtém-se a intensidade da freqüência correspondente ao filtro na posição de cada píxel. Diferentemente da expansão de coeficientes, a projeção de coeficientes não pode ser feita de maneira inversa. A Figura 2.5 apresenta resultados da convolução de um imagem com filtros de Gabor. A imagem analisada é uma foto aérea de plantação de eucaliptos. A região onde há plantação fica nitidamente destacada após algumas das filtragens. Isso acontece devido à regularidade de linhas paralelas presentes na região, o que gera uma freqüência quase constante. As regiões de floresta e estradas também ficam destacadas, pois algumas freqüências mais altas tem maior intensidade nessas áreas. Porém, a nitidez é inferior em relação ao primeiro caso. Em situações em que a imagem analisada apresenta linhas aproximadamente paralelas, como impressões digitais, a utilização de filtros de Gabor normalmente apresenta bons resultados (Klimanee e Nguyen, 2004; Lee e Wang, 1999; Xu e Zhang, 2005)

## 2.5 Considerações Finais

Este capítulo apresentou uma revisão sobre diversos métodos de extração de características de texturas. Observa-se que o número total de características é muito grande e alguns dos principais métodos geram uma quantidade arbitrária de características, como os baseados

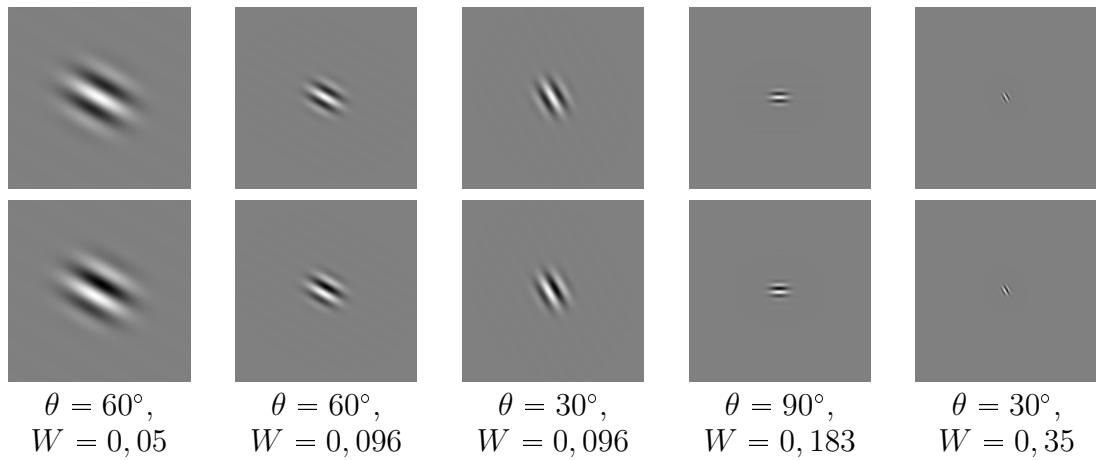


Figura 2.4: Filtros de Gabor no domínio do espaço. Esses filtros foram gerados pela estratégia apresentada utilizando-se  $S = 4$ ,  $K = 6$ ,  $U_l = 0.05$  e  $U_h = 0.35$ . A primeira linha da tabela, de cima para baixo, mostra a parte real dos filtros e a segunda linha, a parte imaginária.

em processamento de sinais. Conseqüentemente, há a necessidade de se avaliar quais são os métodos que geram as características que melhor descrevem as texturas. Além disso, de todas as características obtidas, é interessante também avaliar quais são as mais adequadas para diferentes tipos de imagens. O Capítulo 3 trata justamente de técnicas para essa avaliação, processo chamado de **seleção de características**.

Os experimentos, que demonstram o uso de seleção de características de texturas, são apresentados no Capítulo 5. As características de texturas utilizadas foram as de estatísticas de primeira ordem, de matrizes de co-ocorrência e de filtros de Gabor. Esses métodos foram escolhidos por serem amplamente utilizados e por empregarem abordagens bem distintas. Apesar de dois de esses métodos serem estatísticos, um é de primeira ordem e o outro é de segunda ordem.

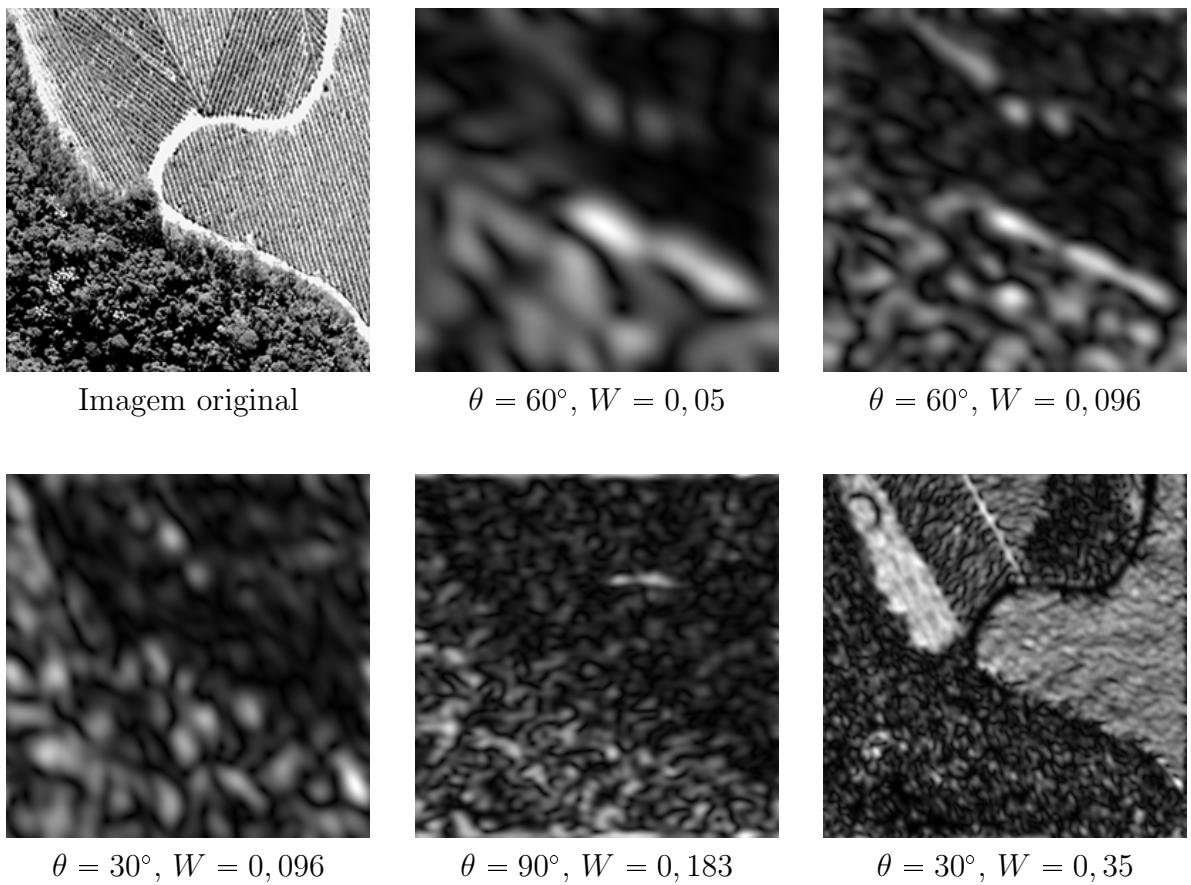


Figura 2.5: Exemplos de filtragens. Os filtros utilizados são os mesmos apresentados na Figura 2.4.

## Seleção de Características

---

---

Como foi apresentado no Capítulo 2, existem diversos métodos para extração de características de texturas de imagens. Essas características podem ser utilizadas em conjunto para classificação, segmentação ou recuperação de imagens por conteúdo (*content-based image retrieval*, CBIR). Alguns desses métodos extraem características mais relevantes para determinados tipos de imagens, enquanto as características obtidas por outros métodos podem atrapalhar a classificação. Além disso, a utilização de diferentes métodos leva a uma classificação mais precisa se comparada à utilização de métodos isolados (Jain e Zongker, 1997). Um processo de seleção pode ser utilizado para se determinar quais características obtidas por quais métodos são mais relevantes para determinado tipo de imagem.

A redução do número de características, além de possivelmente melhorar a precisão da classificação, diminui o custo computacional tanto da extração de características como da classificação. A princípio, quando se aumenta o número de características, a classificação se torna mais precisa, pois a distinção entre os exemplos a serem classificados fica maior. Porém, em algumas situações, a precisão pode cair depois de determinado ponto desse aumento. Isso se deve à **maldição da dimensionalidade**, termo criado por Bellman (1961) e que relaciona o número de exemplos com o número de dimensões em uma base de dados. Mantendo-se constante o número de exemplos e aumentando-se o número de dimensões, a distância entre esses exemplos aumenta exponencialmente. Quanto mais esparsos estão os padrões, maior a dificuldade de treinamento para certos algoritmos de classificação.

A **seleção de características** é um problema de otimização. Dado um conjunto de características  $Y = \{1, 2, \dots, D\}$ , um subconjunto que melhor atinge certo objetivo deve ser encontrado. O objetivo está relacionado com a maximização de uma **função**

**critério**  $J(\cdot)$ , que mede o grau de eficiência do subconjunto. Kudo e Sklansky (2000) dividem os objetivos em três tipos. No **objetivo tipo A**, a função  $J(\cdot)$  deve ser maximizada para um subconjunto de  $d$  características. No **objetivo tipo B**, deve ser encontrado o menor subconjunto para que  $J(\cdot)$  não seja menor que um valor especificado. E o **objetivo tipo C** é a combinação de A e B, ou seja, procura-se minimizar o tamanho do subconjunto e maximizar  $J(\cdot)$ .

A função critério pode ser **dependente** ou **independente** de um algoritmo de reconhecimento de padrões(Liu e Yu, 2005). Uma função critério dependente avalia as características ou os subconjuntos de características estimando a precisão um classificador, por exemplo. As funções critério independentes normalmente utilizam medidas estatísticas não vinculadas diretamente a algoritmos de reconhecimento de padrões e são **monotônicas**. O valor de uma função critério monotônica nunca decresce com o acréscimo de uma ou mais características. As funções critério **dependentes** em geral não são monotônicas.

Diversas revisões e comparações de algoritmos de seleção de características já foram publicadas (Liu e Yu, 2005; Kudo e Sklansky, 2000; Jain e Zongker, 1997; Ferri et al., 1994). A Figura 3.1 mostra uma adaptação da taxonomia dos algoritmos de seleção de características apresentada por Jain e Zongker (1997).

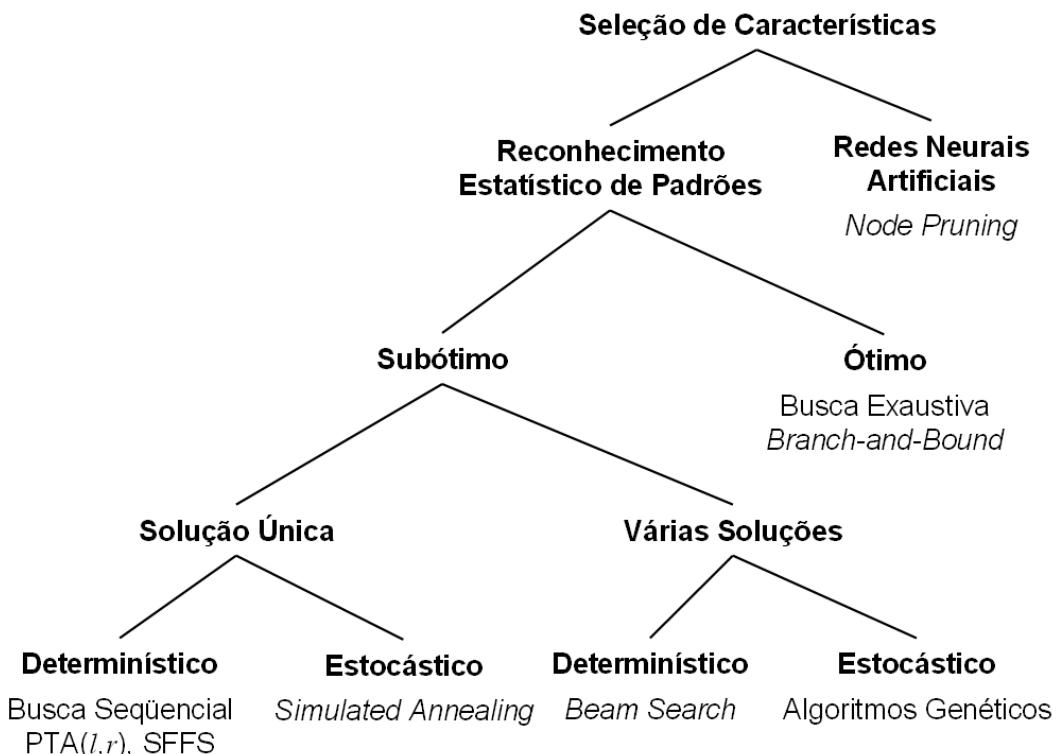


Figura 3.1: Taxonomia dos algoritmos de seleção de características (Jain e Zongker, 1997).

Uma solução **ótima** é aquela que certamente é a melhor possível, enquanto que **subótima** não possui essa garantia. Portanto, nesse segundo caso, pode ser avaliado o

quão próximo o algoritmo é capaz de chegar da solução ótima, além do custo computacional. Na abordagem “**solução única**”, apenas um subconjunto de características é mantido e modificado pelo algoritmo. Na abordagem “**várias soluções**”, os algoritmos trabalham com uma população ou conjunto de subconjuntos ao mesmo tempo. Algoritmos **determinísticos** são aqueles que chegam sempre à mesma solução para uma certa entrada. Já os **estocásticos** realizam operações aleatórias, o que pode levar a diferentes soluções. O valor da função critério também pode apresentar comportamento estocástico causado por alguns algoritmos de classificação.

A **busca exaustiva** avalia todos os possíveis subconjuntos de características e, consequentemente, é capaz de encontrar a solução ótima. Porém, o número total de chamadas à função critério é extremamente alto, o que torna o algoritmo inviável em muitos casos. É difícil determinar as situações em que um método é melhor do que outro. Considerando apenas o objetivo tipo A, a busca exaustiva apresenta bom desempenho quando  $d = 1$ , já que apenas  $D$  chamadas da função critério são realizadas. Quando  $2 \leq d < 6$ , a busca exaustiva ainda pode apresentar um desempenho satisfatório. Quanto mais próximo  $d$  está de  $D/2$ , maior o número de chamadas da função critério. A busca exaustiva normalmente é empregada na avaliação de algoritmos de seleção não ótimos quando é utilizada uma função critério não monotônica. Como é apresentado a seguir, o *branch and bound*, que também é capaz de encontrar a solução ótima e tem melhor desempenho, não pode ser aplicado nesse caso pois é restrito a funções critério monotônicas.

O método ótimo tradicionalmente utilizado para seleção de características é o ***branch and bound***(Narendra e Fukunaga, 1977). Esse método utiliza uma árvore de busca em que as folhas representam todos os subconjuntos possíveis de tamanho  $d$ . Pelo fato da função critério ser monotônica, não é necessário percorrer a árvore toda para encontrar a solução ótima. Diversas melhorias já foram propostas desde a versão original(Yu e Yuan, 1993; Somol et al., 2000, 2001; Chen, 2003; Nakariyakul e Casasent, 2007). Apesar de ser muito mais eficiente do que a busca exaustiva em muitos casos, o *branch and bound* ainda é custoso computacionalmente e é utilizado apenas quando a solução ótima realmente é desejada. Uma explicação mais detalhada sobre o *branch and bound* é encontrada na Seção 3.2 e uma nova estratégia para melhorar a eficiência do algoritmo é apresentada no Capítulo 4.

Os métodos agrupados na taxonomia como determinísticos de solução única são chamados nessa dissertação de **busca seqüencial** (Kittler, 1978 apud Kudo e Sklansky, 2000; Pudil et al., 1994). Os mais simples são o *sequential forward selection* (SFS) e o *sequential backward selection* (SBS). Neles, uma característica é adicionada (ou removida) ao subconjunto da iteração anterior em cada iteração. A escolha da característica é feita pelo valor de  $J(\cdot)$ . O processo é repetido até que o subconjunto atinja o tamanho  $d$ . Portanto, o objetivo é do tipo A. Um algoritmo mais eficiente, chamado *plus-l take-away-r* (PTA( $l, r$ )), adiciona  $l$  características e remove  $r$  características em cada ite-

ração, sendo que  $l \neq r$ . Porém, é difícil determinar os melhores valores para  $l$  e  $r$  antes da execução do algoritmo. Os algoritmos *sequential floating forward selection* (SFFS) e *sequential floating backward selection* (SBFS) foram propostos para ajudar a resolver esse problema (Pudil et al., 1994). Parecido com o PTA( $l, r$ ), em cada iteração do SFFS uma característica é adicionada e zero ou mais características são removidas enquanto forem encontrados subconjuntos melhores. O SBFS é análogo ao SFFS. Esses métodos são explicados detalhadamente na Seção 3.3.

**Beam search** e **simulated annealing**, presentes na taxonomia da Figura 3.1, são citados por Jain e Zongker (1997) mas não são incluídos nas comparações realizadas por eles. Beam search é uma modificação do *best-first search*. O problema é tratado como um grafo, em que cada nó corresponde a um subconjunto de características.

O nome *simulated annealing* vem do processo de fabricação de alguns materiais pelo controle da temperatura. É um algoritmo de otimização, assim como outros para seleção de características. Nesse caso, o objetivo é achar o mínimo de uma função. Em cada iteração, o valor de  $s$  é alterado aleatoriamente e o valor de  $T$ , relacionado à temperatura, é decrementado. Se o valor da função em  $s$  for menor que na iteração anterior, o novo valor é aceito. Caso contrário, o valor pode ser aceito com certa probabilidade, que diminui conforme a temperatura é reduzida. Dessa maneira, o valor da função em  $s$  pode aumentar ou diminuir, permitindo que a solução não fique presa em um mínimo local.

**Algoritmos genéticos** são amplamente empregados em diversos problemas de otimização e busca. Um conjunto de possíveis soluções é tratado com uma população de cromossomos. Em seleção de características, em geral, os cromossomos correspondem a seqüências binárias, que representam subconjuntos de características. Em cada geração, alguns cromossomos são substituídos por outros. Os novos cromossomos são criados por cruzamentos e mutações dos demais. A escolha dos indivíduos que serão eliminados é feita de acordo com uma função objetivo, que favorece os subconjuntos com poucas características e com valores altos de  $J(\cdot)$ . O objetivo desse método é do tipo C. Mais detalhes sobre algoritmos genéticos e a sua aplicação em seleção de características são apresentados na Seção 3.5.

**Redes neurais artificiais** podem ser empregadas em diversos problemas de reconhecimento de padrões. O funcionamento foi originalmente inspirado em redes neurais naturais. Atualmente existe uma grande quantidade de tipos de redes neurais e algoritmos de treinamento. Em seleção de características, as redes neurais são utilizadas de maneira diferente dos outros métodos apresentados neste trabalho, que realizam a maximização de uma função. Nesse caso, o método de seleção analisa uma rede neural treinada e determina a relevância das características, que são selecionadas a partir disso. Existem abordagens semelhantes que utilizam outros algoritmos, como árvores de decisão (Kohavi e John, 1997). Uma revisão sobre redes neurais e a sua aplicação em seleção de características são apresentadas na seção 3.4.

### 3.1 Função Critério

Uma função critério  $J(\cdot)$  mede a “qualidade” de um subconjunto de características. Normalmente, uma função critério tem como entrada um conjunto de números inteiros, que representam as características, e retorna um valor real. Nesta dissertação, assume-se que quanto maior o valor retornado pela função critério, melhor o subconjunto de características.

Em geral, as funções critério não são equivalentes. Isto é, um subconjunto com valor alto para uma função critério pode apresentar valor baixo quando outra função é utilizada. Portanto, se um subconjunto é ótimo em relação a determinada função critério, não significa que será ótimo em relação a outras funções. Para que um subconjunto seja avaliado pela função critério, uma base de dados é empregada. Logo, a avaliação é vinculada à base de dados. A utilização de uma base de dados que não representa bem o problema abordado prejudica muito um processo de seleção de características.

Cada função critério avalia certos aspectos dos subconjuntos. Algumas funções são **independentes** dos algoritmos utilizados em etapas seguintes do reconhecimento de padrões(Liu e Yu, 2005). As medidas estatísticas de distância são independentes. Algumas dessas medidas são a distância de Mahalanobis, a distância de Bhattacharyya e a distância de Jeffries-Matusita. A distância corresponde ao nível de separação entre os exemplos de cada classe, assumindo que há apenas um *cluster* para cada classe na base de dados.

Uma função critério **dependente** faz a avaliação com base em um algoritmo específico de reconhecimento de padrões. Em geral, a taxa de acerto de um classificador é utilizada nesse caso. Existem diferentes métodos para estimar a precisão de um classificador, isto é, calcular a taxa de acerto. O *holdout* (Resende, 2003) consiste na divisão aleatória da base de dados em dois conjuntos, um de treino e outro de teste, sendo que não pode haver repetição de exemplos nos conjuntos. Normalmente 70% dos exemplos formam o conjunto de treino e os 30% restante, o conjunto de teste. O classificador é então treinado com o conjunto de treino e a taxa de acerto é estimada pela avaliação do resultado obtido com o conjunto de teste. O *holdout* é um método simples, mas que não apresenta boa exatidão. A realização de testes diferentes com a mesma base de dados permite uma avaliação melhor. Para isso, pode ser utilizado o *k-fold cross-validation* (Resende, 2003), que consiste na divisão aleatória da base de dados em  $k$  partições de aproximadamente o mesmo tamanho, também sem repetição de exemplos. São realizados  $k$  testes, sendo que, em cada um, uma das partições é utilizada como conjunto de teste e as restantes formam o conjunto de treino. Normalmente é utilizado o *10-fold cross-validation*. Para que a estimativa seja mais exata, o método pode ser repetido  $n$  vezes (Kohavi e John, 1997). Assim, em cada uma das  $n$  repetições, todo o processo de um *k-fold cross-validation* é realizado:  $k$  partições são geradas e  $k$  testes são feitos. A precisão é estimada calculando-se a média dos  $n \cdot k$  testes.

As características selecionadas com uma função critério dependente são específicas para o algoritmo de reconhecimento de padrões correspondente. Assim, os resultados tendem a ser melhores do que se uma função critério independente é utilizada. Porém, em algumas situações, uma função critério independente pode ser necessária. Por exemplo, quando o algoritmo não é conhecido ou é lento.

Em geral, uma função critério independente é **monotônica**, enquanto que uma função critério dependente não é. Uma função  $J(\cdot)$  é monotônica se, dados os subconjuntos  $X_1$  e  $X_2$ , sendo que  $X_1 \subset X_2$ , então  $J(X_1) \leq J(X_2)$ .

A seguir são detalhadas a distância de Bhattacharyya, amplamente utilizada quando há duas classes; a distância de Jeffries-Matusita, que é uma medida entre 0 e 2 e pode ser utilizada para qualquer número de classes; e o classificador de distância mínima, que é conveniente para uso em uma função critério dependente por ser rápido.

### 3.1.1 Distância de Bhattacharyya

A distância de Bhattacharyya (Fukunaga, 1990) entre as classes  $i$  e  $j$  para distribuição normal é definida por

$$B_{ij} = \frac{1}{8} \cdot (\mathbf{M}_j - \mathbf{M}_i)^T \cdot \left( \frac{\mathbf{C}_i + \mathbf{C}_j}{2} \right)^{-1} \cdot (\mathbf{M}_j - \mathbf{M}_i) + \frac{1}{2} \ln \left( \frac{\left| \frac{\mathbf{C}_i + \mathbf{C}_j}{2} \right|}{\sqrt{|\mathbf{C}_j||\mathbf{C}_i|}} \right), \quad (3.1)$$

sendo que

$\mathbf{M}_i$  e  $\mathbf{M}_j$  são os vetores de médias das classes  $i$  e  $j$ , respectivamente,

$\mathbf{C}_i$  e  $\mathbf{C}_j$  são as matrizes de covariância das classes  $i$  e  $j$ , respectivamente,

$|\cdot|$  representa o determinante da matriz.

A Equação 3.1 é formada pela soma de dois termos. O primeiro termo é a distância de Mahalanobis entre os centróides das duas classes. Os centróides são equivalentes aos vetores de médias. Caso  $\mathbf{M}_i = \mathbf{M}_j$ , o primeiro termo é reduzido a zero. O segundo termo corresponde à diferença entre as covariâncias das classes. Caso  $\mathbf{C}_i = \mathbf{C}_j$ , o segundo termo é reduzido a zero. Portanto, a diferença entre as classes é medida em relação às diferenças das médias e das covariâncias.

Algumas estratégias podem ser usadas para melhorar o desempenho do cálculo da distância de Bhattacharyya quando esta é utilizada em seleção de características. Os elementos de  $\mathbf{M}_i$ ,  $\mathbf{M}_j$ ,  $\mathbf{C}_i$  e  $\mathbf{C}_j$  para um subconjunto de características estão presentes nos vetores de médias e matrizes de covariância do conjunto completo de características. O mesmo vale para a diferença dos vetores de médias ( $\mathbf{M}_j - \mathbf{M}_i$ ) e a média das matrizes de

covariância ( $\bar{\mathbf{C}}_{ij} = (\mathbf{C}_i + \mathbf{C}_j)/2$ ). Logo, esses cálculos podem ser realizados apenas uma vez e reaproveitados em todas as chamadas de  $J(\cdot)$ .

A matriz de covariância é simétrica e semidefinida positiva. Se nenhuma característica for linearmente dependente de outra, então é definida positiva. A média das matrizes de covariância preserva essas propriedades. Assim, a maneira mais eficiente de se encontrar a matriz inversa e o determinante das matrizes de covariância é utilizando a decomposição de Cholesky (Press et al., 1992). Se alguma característica for linearmente dependente de outra, então o determinante da matriz de covariância é zero e a inversão não pode ser realizada. Isso acontece se a base de dados contiver mais características do que exemplos para alguma classe ou se alguma característica for constante. Tal situação impede o cálculo da distância de Bhattacharyya.

Como o determinante de uma matriz de covariância muito grande costuma resultar num valor muito pequeno, o cálculo da distância de Bhattacharyya pode causar erro de ponto flutuante. Uma estratégia para resolver esse problema é a mudança na ordem em que as operações são realizadas. O cálculo deve então ser feito por

$$B_{ij} = \frac{1}{8} \cdot (\mathbf{M}_j - \mathbf{M}_i)^T \cdot \bar{\mathbf{C}}_{ij}^{-1} \cdot (\mathbf{M}_j - \mathbf{M}_i) + \frac{1}{2} \ln \left( \frac{\sqrt{|\bar{\mathbf{C}}_{ij}|}}{\sqrt{|\mathbf{C}_i|}} \cdot \frac{\sqrt{|\bar{\mathbf{C}}_{ij}|}}{\sqrt{|\mathbf{C}_j|}} \right). \quad (3.2)$$

### 3.1.2 Distância de Jeffries-Matusita

A distância de Jeffries-Matusita (JM) (Richards, 1993) entre as classes  $i$  e  $j$  para distribuição normal é definida por

$$J_{ij} = 2(1 - \exp(-B_{ij})). \quad (3.3)$$

O valor de  $J_{ij}$  varia entre 0 e 2, sendo que a distância 2 significa uma separação completa dos exemplos entre as classes, considerando-se que a distribuição é normal e que há apenas duas classes. Caso a probabilidade *a priori* seja a mesma para todas as classes, a função critério pode ser definida como a média entre a distância de Jeffries-Matusita de todos os pares de classes.

### 3.1.3 Classificador de Distância Mínima

Para a utilização de uma função critério dependente, há a necessidade de um algoritmo de classificação rápido, já que a função é chamada muitas vezes durante a seleção de características. Uma boa opção é o Classificador de Distância Mínima (CDM) (Gonzalez e Woods, 1992; Richards, 1993), pois além de simples e rápido, é bem adaptável para seleção de características. Essa adaptação é possível pois o treinamento pode ser realizado apenas uma vez para todas as características e reaproveitada na avaliação de qualquer

subconjunto de características.

O CDM utiliza para a classificação os centróides de cada classe do conjunto de treino. Um centróide equivale ao vetor de médias dos exemplos de uma classe. A classe de um exemplo novo é definida como sendo a do centróide mais próximo. A princípio, a distância euclidiana é utilizada pelo CDM. Entretanto, pode-se adotar outras distâncias, como a euclidiana normalizada ou a de Mahalanobis. O treinamento corresponde ao cálculo dos vetores de médias e outras medidas necessárias, caso seja utilizada alguma distância diferente da euclidiana. O processo de classificação é apresentado com formalismo matemático a seguir.

O quadrado da distância euclidiana entre um exemplo  $\mathbf{X}$  e o centróide  $\mathbf{M}_i$  da classe  $i$  é definido como

$$e(\mathbf{X}, \mathbf{M}_i)^2 = (\mathbf{X} - \mathbf{M}_i)^T \cdot (\mathbf{X} - \mathbf{M}_i). \quad (3.4)$$

A classe  $c$  do exemplo  $\mathbf{X}$  é determinada por

$$c = \operatorname{argmin}_{i=1,\dots,N} \left( e(\mathbf{X}, \mathbf{M}_i)^2 \right), \quad (3.5)$$

sendo que  $N$  é o número total de classes. Caso seja utilizada a distância euclidiana normalizada, essa medida é calculada por

$$e'(\mathbf{X}, \mathbf{M}_i)^2 = (\mathbf{X} - \mathbf{M}_i)^T \cdot \mathbf{P}_i^{-1} \cdot (\mathbf{X} - \mathbf{M}_i), \quad (3.6)$$

sendo que  $\mathbf{P}_i$  é uma matriz diagonal com as variâncias das características dos exemplos de treinamento da classe  $i$ . A classe do exemplo  $\mathbf{X}$  é determinada de maneira análoga à apresentada pela Equação 3.5.

## 3.2 Branch and Bound

Para que um subconjunto de características seja considerado ótimo para uma determinada função critério, deve existir a garantia de que todos os outros subconjuntos do mesmo tamanho possuam um valor inferior para tal função. A busca exaustiva faz isso calculando o valor da função critério para todos os subconjuntos. Essa é a maneira mais simples e muitas vezes inviável. Quando a função critério é monotônica, é possível utilizar o *branch and bound*, que encontra o subconjunto ótimo sem precisar avaliar todos os outros.

O *branch and bound* é um algoritmo genérico de otimização. Desde a primeira proposta de utilização do *branch and bound* especificamente para seleção de características (Narendra e Fukunaga, 1977), diversas melhorias foram propostas (Yu e Yuan, 1993; Somol et al., 2000, 2001; Chen, 2003; Nakariyakul e Casasent, 2007). Para facilitar o entendimento, as diferentes versões desse algoritmo serão apresentadas a seguir, da mais

simples para a mais complexa. Cada versão corresponde a alguma versão anterior com uma ou mais estratégias acrescentadas ou substituídas. A versão mais simples é chamada nesta dissertação de ***branch and bound*** **básico** (Narendra e Fukunaga, 1977). Uma propriedade comum a todas as versões do *branch and bound* é o uso de uma árvore de busca (*solution tree*), sendo que as folhas representam todos os subconjuntos de determinado tamanho. A utilização de uma função critério monotônica permite que ramos inteiros da árvore sejam descartados sem prejudicar a busca pelo subconjunto ótimo. A eficiência do algoritmo melhora se as características forem ordenadas de acordo com o valor da função critério durante a construção da árvore. Isso é realizado pelo ***branch and bound*** **ordenado** (Narendra e Fukunaga, 1977). O cálculo da função critério pode ser realizado **recursivamente**, aproveitando-se parte do que foi calculado na etapa anterior (Narendra e Fukunaga, 1977). Essa é uma estratégia importante para diferentes versões do *branch and bound*. O tamanho da árvore pode ser reduzido eliminando-se alguns nós desnecessários, obtendo-se uma **árvore de busca mínima** (*minimum solution tree*) (Yu e Yuan, 1993). O ***branch and bound*** **rápido** (Somol et al., 2000) é uma versão do algoritmo em que são realizadas previsões dos valores da função critério e algumas decisões são tomadas a partir dessas previsões. O ***branch and bound*** **com previsão parcial** (Somol et al., 2001) utiliza a mesma técnica de previsão do *branch and bound* rápido, mas apenas para a ordenação das características. Nas etapas seguintes, os valores reais da função critério são calculados recursivamente. A estratégia de **busca da direita para a esquerda** (Chen, 2003) consiste no armazenamento de informações quando ocorrem podas para que outras podas possam ser realizadas sem a chamada da função critério. O ***branch and bound*** **adaptativo** (Nakariyakul e Casasent, 2007) utiliza diversas estratégias: previsão (diferente da utilizada pelo *branch and bound* rápido), uma única ordenação antes do percurso pela árvore, obtenção de uma solução inicial com outro algoritmo, início do percurso a partir de um determinado nível da árvore e a busca da direita para a esquerda.

A seguir, essas versões e estratégias do *branch and bound* serão explicadas de maneira mais detalhada. No Capítulo 4, uma nova estratégia, chamada de **floresta**, será apresentada. Na estratégia floresta, diversas árvores são utilizadas para a busca pelo subconjunto ótimo.

### 3.2.1 ***Branch and Bound*** básico

O *branch and bound* seleciona um subconjunto de tamanho fixo  $d$  de um conjunto original de tamanho  $D$ . A seleção é realizada percorrendo-se uma árvore de busca. A raiz da árvore representa o conjunto original  $Y$ . Os outros nós representam subconjuntos de  $Y$  e as folhas representam todos subconjuntos possíveis de tamanho  $d$ . A Figura 3.2 mostra um exemplo em que  $D = 6$ ,  $d = 2$  e  $Y = \{1, 2, 3, 4, 5, 6\}$ . O subconjunto de um nó é formado pelo subconjunto do nó pai com uma, e somente uma, característica removida.

O rótulo das arestas na Figura 3.2 representa a característica removida. O subconjunto de características e o valor de  $J(\cdot)$  estão indicados próximo do nó correspondente. Como uma característica é removida por nível no percurso da raiz a uma folha, a árvore possui  $D-d+1$  níveis. O número  $k$  de um nível representa o número de características removidas de  $Y$ . O subconjunto  $X_k$  corresponde a um nó do nível  $k$ . Por exemplo, na Figura 3.2,  $X_2 = \{1, 3, 5, 6\}$  para o nó 7 e  $X_4 = \{1, 2\}$  para o nó 5. A escolha das características que devem ser removidas de cada nó é feita de acordo com algumas regras que serão apresentadas a seguir.

Normalmente o *branch and bound* é implementado de maneira que a árvore é construída conforme é percorrida. Neste trabalho, o *branch and bound* é implementado de maneira recursiva (recursão, nesse caso, não é o mesmo assunto do cálculo recursivo da função critério). A principal etapa da execução do *branch and bound* é chamada de **expansão**, em que um nó é visitado, o subconjunto correspondente é analisado e a expansão dos nós filhos é chamada.

O percurso é realizado a partir da raiz, de cima para baixo e da direita para a esquerda. A numeração interna dos nós da Figura 3.2 mostra o caminho do percurso. Durante a **expansão**, o valor de  $J(X_k)$  é comparado com o valor de um limite  $B$ . Quando o nó analisado é uma folha, se  $J(X_{D-d}) > B$ , então o valor de  $B$  é atualizado,  $B = J(X_{D-d})$ . Assim, a variável  $B$  armazena o maior valor encontrado em uma folha até o momento. Essa atualização é realizada nos nós 5, 9 e 19 da Figura 3.2. Como a função critério é monotônica, o valor de  $J(\cdot)$  nunca aumenta na passagem de um nó para seu sucessor. Portanto, se  $J(X_k) \leq B$ , não há motivo para que a busca continue na subárvore cuja raiz é o nó correspondente ao subconjunto  $X_k$ , pois nenhum valor maior do que  $B$  será encontrado nessa subárvore. Conseqüentemente, esse ramo da árvore pode ser podado e uma quantidade de subconjuntos é eliminada da busca. Assim, o *branch and bound* consegue encontrar o subconjunto ótimo sem precisar avaliar todos. Sempre que o valor de  $B$  é atualizado, o subconjunto correspondente  $X_{D-d}$  deve ser armazenado em  $X'$ . Com isso, quando o algoritmo conclui a busca por toda a árvore, o subconjunto ótimo é  $X'$  e  $J(X') = B$ .

Seja  $F = (f_1, f_2, \dots, f_{D-d})$  a seqüência ordenada das características removidas no caminho da raiz até uma folha. Por exemplo, no caminho da raiz até o nó 9 na Figura 3.2,  $F = (2, 4, 5, 6)$ . Para garantir que não haja repetição de subconjuntos na árvore, a seguinte regra deve ser seguida:

$$f_1 < f_2 < \dots < f_{D-d}. \quad (3.7)$$

Seja  $q_k$  o número de sucessores de um nó do nível  $k$ . A partir da Equação 3.7, conclui-se que o número mais alto da primeira característica que pode ser removida é  $f_1 = d + 1$ . Portanto, a raiz deve possuir  $q_0 = d + 1$  sucessores. Para os demais nós, o número de

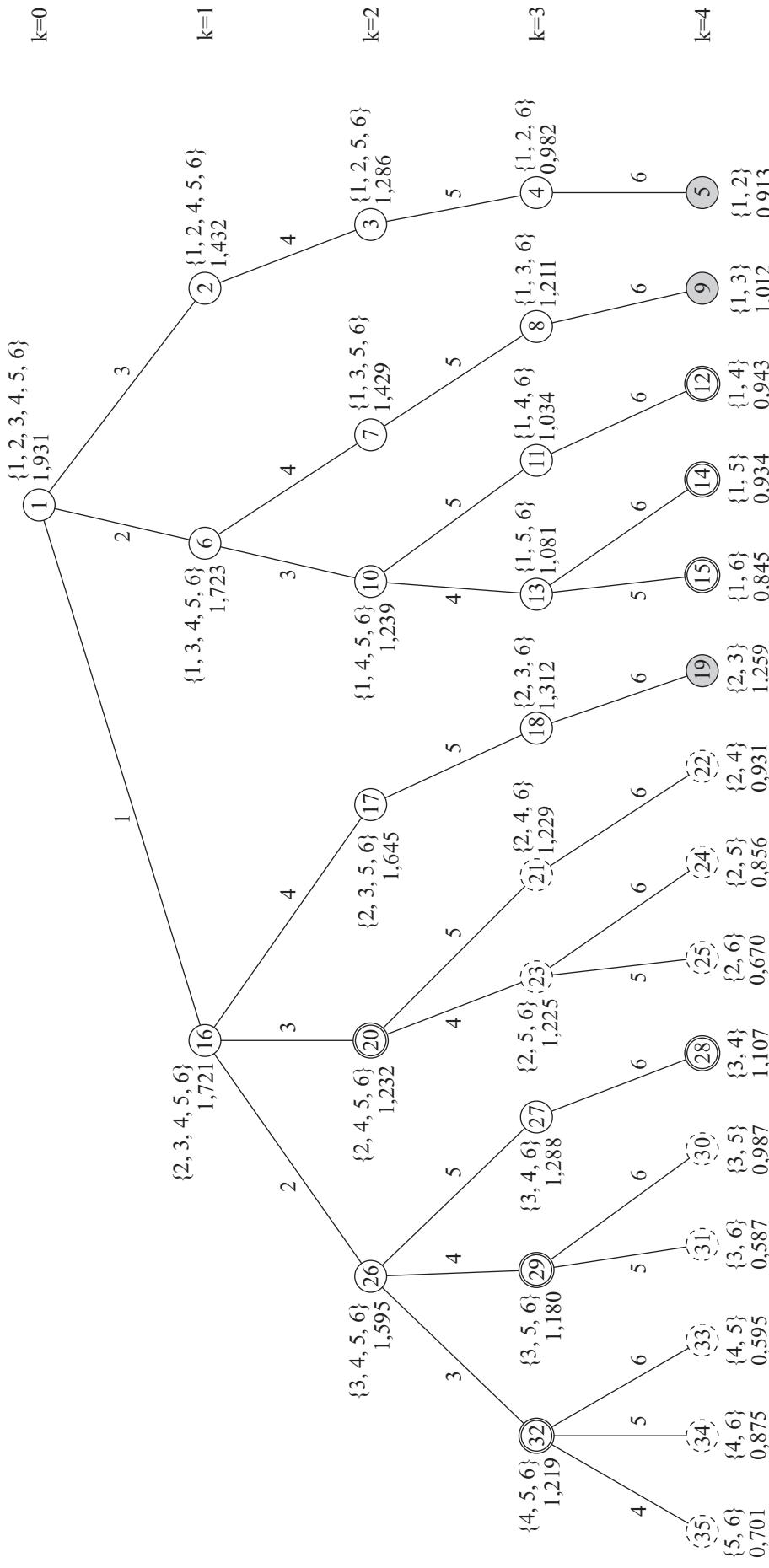


Figura 3.2: Árvore de busca do *branch and bound* básico para  $D = 6$  e  $d = 2$ . A numeração interna dos nós indica o caminho em que o percurso é realizado. O subconjunto de características e o valor de  $J(\cdot)$  estão indicados próximo do nó correspondente. O rótulo de cada aresta indica a característica que foi removida na passagem de um nó do nível  $k$  para um nó do nível  $k+1$ . Os nós preenchidos com cinza indicam que o limite foi atualizado. Os nós com contorno duplo indicam que foi encontrado  $J(\cdot) < B$  e, se o nível do nó for  $k < 4$ , indicam poda. Os nós com contorno tracejado foram eliminados pelas podas.

sucessores é determinado na expansão do nó pai. Os nós sucessores devem receber uma numeração  $p$  da esquerda para direita. Por exemplo, para os sucessores do nó 16 na Figura 3.2,  $p = 1$  para o nó 26,  $p = 2$  para o nó 20 e  $p = 3$  para o nó 17. Assim,  $q_{k+1} = q_k - p + 1$ . Seguindo-se essas regras, todos os subconjuntos de tamanho  $d$  são representados sem repetição nas folhas.

### 3.2.2 *Branch and Bound* ordenado

Quanto menor o valor de  $J(\cdot)$  para nós mais a esquerda da árvore, maior o número de nós eliminados por podas, pois o número de ramificações aumenta da direita para a esquerda. Além disso, quanto menor o valor de  $J(\cdot)$  em qualquer nó, maior a probabilidade de poda. A mudança na ordem com que as características são removidas pode ser utilizada para se conseguir valores mais baixos na esquerda da árvore. O *branch and bound* ordenado utiliza essa estratégia reordenando as características em cada expansão de um nó. A Figura 3.3 mostra o mesmo problema apresentado na Figura 3.2, mas utilizando uma árvore de busca do *branch and bound* ordenado.

Para que as regras apresentadas na seção 3.2.1 sejam seguidas e a reordenação seja possível, as características devem ser representadas como variáveis. O conjunto de todas as características é representado agora como a seqüência ordenada  $Y' = (y_1, y_2, \dots, y_D)$ , sendo que  $1 \leq y_i \leq D$  e  $y_i \neq y_j$  se  $i \neq j$ . As características removidas no caminho da raiz até uma folha são  $F' = (y_{f_1}, y_{f_2}, \dots, y_{f_{D-d}})$ . Assim, a Equação 3.7 ainda é válida para a construção da árvore. Para cada nó, há um conjunto de características disponíveis para remoção,  $T = \{y_a, y_{a+1}, \dots, y_D\}$ , sendo que  $a$  é determinado de acordo com a Equação 3.7. Por exemplo, na Figura 3.3,  $a = 1$  para o nó 1,  $a = 3$  para o nó 6 e  $a = 4$  para o nó 10. Esses valores de  $a$  são coincidentes com o menor número das características que são removidas dos nós da Figura 3.2. Na expansão de um nó, os elementos de  $Y'$  pertencentes a  $T$  são ordenados de maneira crescente de acordo com o valor de  $J(X_k \setminus \{y_i\})$  correspondente. No exemplo, os valores calculados na expansão da raiz são

$$\begin{aligned} J(Y \setminus \{y_1\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{1\}) = J(\{2, 3, 4, 5, 6\}) = 1,721, \\ J(Y \setminus \{y_2\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{2\}) = J(\{1, 3, 4, 5, 6\}) = 1,723, \\ J(Y \setminus \{y_3\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{3\}) = J(\{1, 2, 4, 5, 6\}) = 1,432, \\ J(Y \setminus \{y_4\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{4\}) = J(\{1, 2, 3, 5, 6\}) = 1,847, \\ J(Y \setminus \{y_5\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{5\}) = J(\{1, 2, 3, 4, 6\}) = 1,787, \\ J(Y \setminus \{y_6\}) &= J(\{1, 2, 3, 4, 5, 6\} \setminus \{6\}) = J(\{1, 2, 3, 4, 5\}) = 1,833. \end{aligned}$$

A ordenação passa então a ser  $Y' = (3, 1, 2, 5, 6, 4)$ . As características escolhidas para serem removidas da raiz são 3, 1 e 2, nessa ordem. Os subconjuntos obtidos com a remoção das características 4, 5 e 6 da raiz não são aproveitados na árvore, apesar de

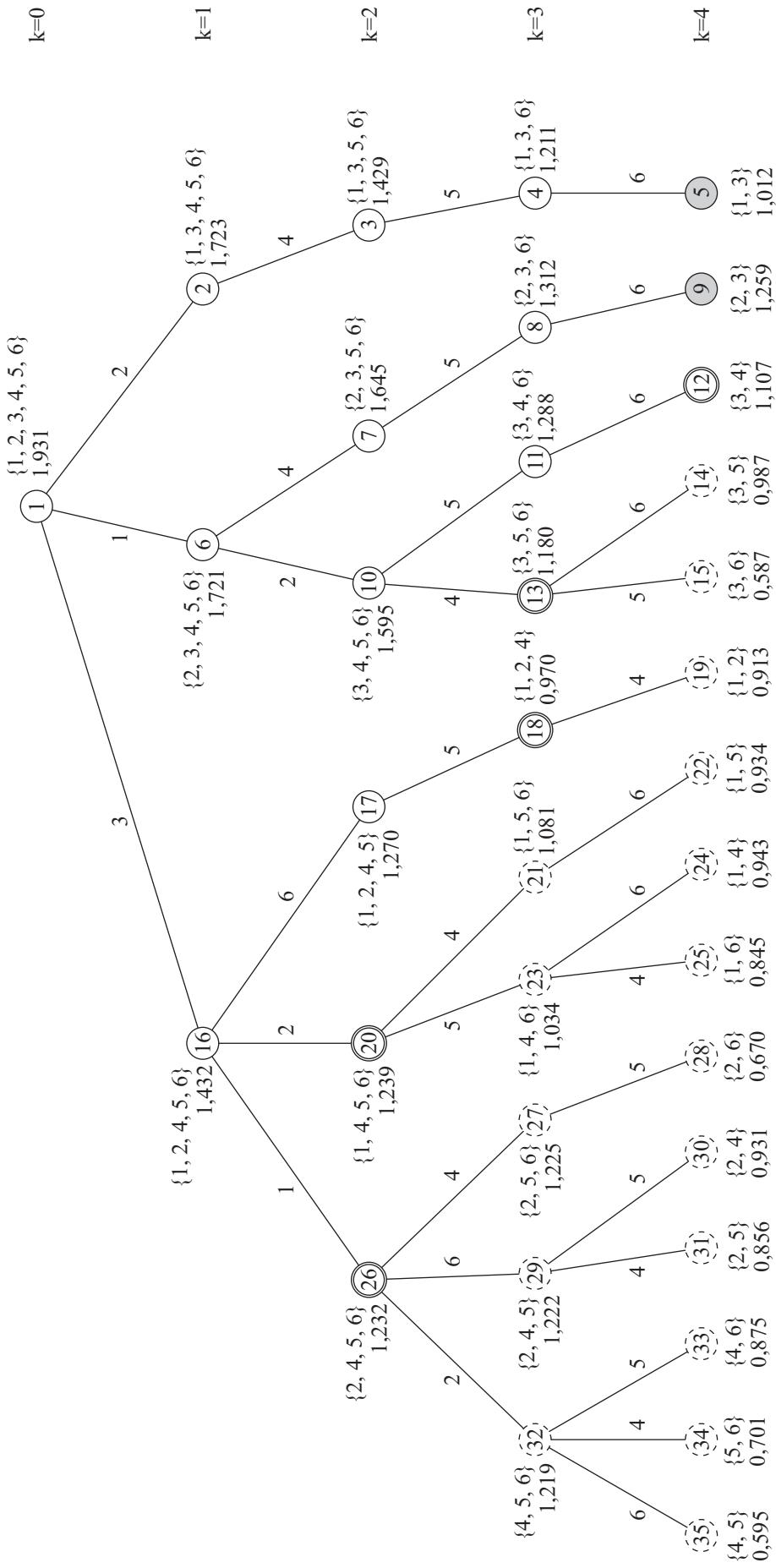


Figura 3.3: Árvore de busca do *branch and bound* ordenado para  $D = 6$  e  $d = 2$ . As notações utilizadas nessa figura são as mesmas da Figura 3.2.

terem sido avaliados. Na expansão do nó 6, os valores calculados são

$$J(X_1 \setminus \{y_3\}) = J(\{2, 3, 4, 5, 6\} \setminus \{2\}) = J(\{3, 4, 5, 6\}) = 1,595,$$

$$J(X_1 \setminus \{y_4\}) = J(\{2, 3, 4, 5, 6\} \setminus \{5\}) = J(\{2, 3, 4, 6\}) = 1,685,$$

$$J(X_1 \setminus \{y_5\}) = J(\{2, 3, 4, 5, 6\} \setminus \{6\}) = J(\{2, 3, 4, 5\}) = 1,702,$$

$$J(X_1 \setminus \{y_6\}) = J(\{2, 3, 4, 5, 6\} \setminus \{4\}) = J(\{2, 3, 5, 6\}) = 1,645.$$

A ordenação resultante é  $Y' = (3, 1, 2, 4, 5, 6)$ . As características escolhidas para serem removidas do nó 6 são 2 e 4, nessa ordem.

Observa-se uma quantidade maior de nós eliminados com as podas na árvore da Figura 3.3 em comparação com aqueles observados na Figura 3.2. Além disso, a ordenação aumenta a probabilidade de encontrar valores mais altos para  $B$  mais cedo. Porém, um número maior de chamadas da função critério é realizado em cada expansão. Na prática, o *branch and bound* ordenado apresenta eficiência superior ao *branch and bound* básico na maioria dos casos. Algumas estratégias já foram propostas para que a ordenação seja realizada sem o grande aumento do número de chamadas da função critério.

### 3.2.3 Cálculo Recursivo do Valor da Função Critério

No percurso pela árvore de busca, o *branch and bound* chama a função critério para um subconjunto com uma característica removida de um outro subconjunto na maioria dos nós. O cálculo do valor da função critério pode ser feito recursivamente a partir de um cálculo realizado anteriormente, reduzindo consideravelmente o esforço computacional. No caso do *branch and bound* básico e do *branch and bound* ordenado, a função critério pode ser calculada sem recursão para a raiz e recursivamente para todos os outros nós.

Esta seção apresenta o método para calcular recursivamente o valor da distância de Bhattacharyya quando a última característica é removida. O processo pode ser estendido para outras medidas de distância que possuam fórmula semelhante. Quando a remoção não é da última característica, o método pode ser adaptado mudando-se a ordem das características. Na matriz de covariância, isso é feito trocando-se as linhas e colunas correspondentes.

Como mostrado na Equação 3.2,  $\bar{\mathbf{C}}_{ij}$  é a média das matrizes de covariância. Seja  $m$  o número de linhas e colunas de  $\bar{\mathbf{C}}_{ij}$ . A  $m$ -ésima linha e a  $m$ -ésima coluna da matriz  $\bar{\mathbf{C}}_{ij}$  e da inversa  $\bar{\mathbf{C}}_{ij}^{-1}$  podem ser destacadas por

$$\bar{\mathbf{C}}_{ij} = \begin{bmatrix} \bar{\mathbf{S}}_{ij} & \mathbf{E} \\ \mathbf{E}^T & s \end{bmatrix}, \quad \bar{\mathbf{C}}_{ij}^{-1} = \begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{G}^T & g \end{bmatrix}.$$

$\bar{\mathbf{S}}_{ij}$  é a media das matrizes de covariância após a remoção da última característica de  $\bar{\mathbf{C}}_{ij}$ .

O determinante e a inversão de  $\bar{\mathbf{S}}_{ij}$  podem ser calculados por

$$|\bar{\mathbf{S}}_{ij}| = \frac{|\bar{\mathbf{C}}_{ij}|}{s - \mathbf{E}^T \cdot \bar{\mathbf{S}}_{ij}^{-1} \cdot \mathbf{E}}, \quad \bar{\mathbf{S}}_{ij}^{-1} = \mathbf{F} - \frac{\mathbf{G} \cdot \mathbf{G}^T}{g}.$$

A multiplicação presente no primeiro termo da fórmula da distância de Bhattacharyya pode ser calculada recursivamente por

$$\begin{aligned} (\mathbf{V}_j - \mathbf{V}_i)^T \cdot \bar{\mathbf{S}}_{ij}^{-1} \cdot (\mathbf{V}_j - \mathbf{V}_i) &= (\mathbf{M}_j - \mathbf{M}_i)^T \cdot \bar{\mathbf{C}}_{ij}^{-1} \cdot (\mathbf{M}_j - \mathbf{M}_i) \\ &\quad - \frac{1}{g} \cdot \left( \begin{bmatrix} \mathbf{G}^T & g \end{bmatrix} \cdot (\mathbf{M}_j - \mathbf{M}_i) \right)^2. \end{aligned}$$

Sendo que  $\mathbf{V}_i$  e  $\mathbf{V}_j$  correspondem aos vetores  $\mathbf{M}_i$  e  $\mathbf{M}_j$  com a última característica removida.

### 3.2.4 Árvore de Busca Mínima

Alguns nós da árvore de busca possuem apenas um sucessor. Por exemplo, os nós 2, 3, 4, 7, e 8 da Figura 3.3. Quando a poda ocorre na expansão desses nós, apenas uma folha é eliminada, como acontece no nó 18. Portanto, é vantajoso não chamar a função critério para esses nós e avançar a busca direto para a folha seguinte. Com a omissão desses nós, obtém-se uma árvore de busca mínima. Além da redução do número de nós, o cálculo da função critério passa a ser mais rápido por ser realizado para subconjuntos menores nesses casos. Porém, a seqüência de chamadas da função critério é interrompida e a avaliação nessas folhas não pode ser feita recursivamente.

### 3.2.5 *Branch and Bound* Rápido

O *branch and bound* rápido é uma modificação do *branch and bound* ordenado que utiliza a árvore de busca mínima e realiza previsões para diminuir o número de chamadas da função critério. A Figura 3.4 mostra um exemplo da árvore desta versão do *branch and bound*, utilizando o mesmo problema das figuras anteriores. Inicialmente, o algoritmo determina quanto cada característica contribui para reduzir o valor de  $J(\cdot)$  quando removida de um subconjunto. Formalmente, isso é descrito pelas equações

$$A_{y_i} = \frac{A_{y_i} \cdot S_{y_i} + J(X_k) - J(X_k \setminus \{y_i\})}{S_{y_i} + 1}, \tag{3.8}$$

$$S_{y_i} = S_{y_i} + 1, \tag{3.9}$$

Os valores de  $A_{y_i}$  e  $S_{y_i}$  são atualizados sempre que os valores reais da função critério são calculados para um nó e para os subconjuntos gerados a partir desse nó.  $A_{y_i}$  representa a contribuição da característica  $y_i$ .  $S_{y_i}$  é o número de atualizações de  $A_{y_i}$  e deve ser

inicializado com  $S_{y_i} = 0$  para  $i = 1, 2, \dots, D$ . Na expansão do nó 1 da Figura 3.4, o valor de  $A_{y_i}$  para todas as características é atualizado, resultando em

$$A_{y_1} = A_1 = \frac{A_1 \cdot S_1 + J(Y) - J(Y \setminus \{1\})}{S_1 + 1} = \frac{A_1 \cdot 0 + 1,931 - 1,721}{0 + 1} = 0,210,$$

$$A_{y_2} = A_2 = \frac{A_2 \cdot S_2 + J(Y) - J(Y \setminus \{2\})}{S_2 + 1} = \frac{A_2 \cdot 0 + 1,931 - 1,723}{0 + 1} = 0,208,$$

$$A_{y_3} = A_3 = \frac{A_3 \cdot S_3 + J(Y) - J(Y \setminus \{3\})}{S_3 + 1} = \frac{A_3 \cdot 0 + 1,931 - 1,432}{0 + 1} = 0,499,$$

$$A_{y_4} = A_2 = \frac{A_4 \cdot S_4 + J(Y) - J(Y \setminus \{4\})}{S_4 + 1} = \frac{A_4 \cdot 0 + 1,931 - 1,847}{0 + 1} = 0,084,$$

$$A_{y_5} = A_5 = \frac{A_5 \cdot S_5 + J(Y) - J(Y \setminus \{5\})}{S_5 + 1} = \frac{A_5 \cdot 0 + 1,931 - 1,787}{0 + 1} = 0,144,$$

$$A_{y_6} = A_6 = \frac{A_6 \cdot S_6 + J(Y) - J(Y \setminus \{6\})}{S_6 + 1} = \frac{A_6 \cdot 0 + 1,931 - 1,833}{0 + 1} = 0,098.$$

O valor previsto  $\hat{J}(\cdot)$  é inicialmente calculado por

$$\hat{J}(X_k \setminus \{y_i\}) = \hat{J}(X_k) - A_{y_i}, \quad (3.10)$$

ou

$$\hat{J}(X_k \setminus \{y_i\}) = J(X_k) - A_{y_i}. \quad (3.11)$$

A Equação 3.10 é empregada quando o valor da função critério foi previsto,  $\hat{J}(X_k)$ , e a equação 3.11 é empregada quando o valor real foi calculado,  $J(X_k)$ . A previsão é permitida apenas quando a contribuição foi atualizada um determinado número de vezes. Isso é definido pelo parâmetro  $\delta$ . Ou seja, a previsão  $\hat{J}(X_k \setminus \{y_i\})$  é realizada apenas quando  $S_{y_i} \geq \delta$ . A ordenação das características é feita de acordo com os valores de  $J(\cdot)$  ou  $\hat{J}(\cdot)$  obtidos até então. Na ordenação realizada na expansão do nó 1 da Figura 3.4, todos os valores de  $J(\cdot)$  utilizados foram reais, pois nenhuma previsão era permitida. Enquanto que na expansão do nó 6, a ordenação foi realizada com todos os valores previstos  $\hat{J}(\cdot)$ .

Para as etapas seguintes, os valores previstos dos nós que farão parte da árvore são alterados para

$$\hat{J}(X_k \setminus \{y_i\}) = \hat{J}(X_k) - \gamma \cdot A_{y_i}, \quad (3.12)$$

ou

$$\hat{J}(X_k \setminus \{y_i\}) = J(X_k) - \gamma \cdot A_{y_i}. \quad (3.13)$$

sendo que  $\gamma$  é um parâmetro do algoritmo para definir o grau de otimismo da previsão. Quando o algoritmo verifica que ocorrerá uma poda a partir de um valor previsto  $\hat{J}(X_k)$ , o valor real  $J(X_k)$  é calculado para substituir o valor previsto e a ocorrência de poda

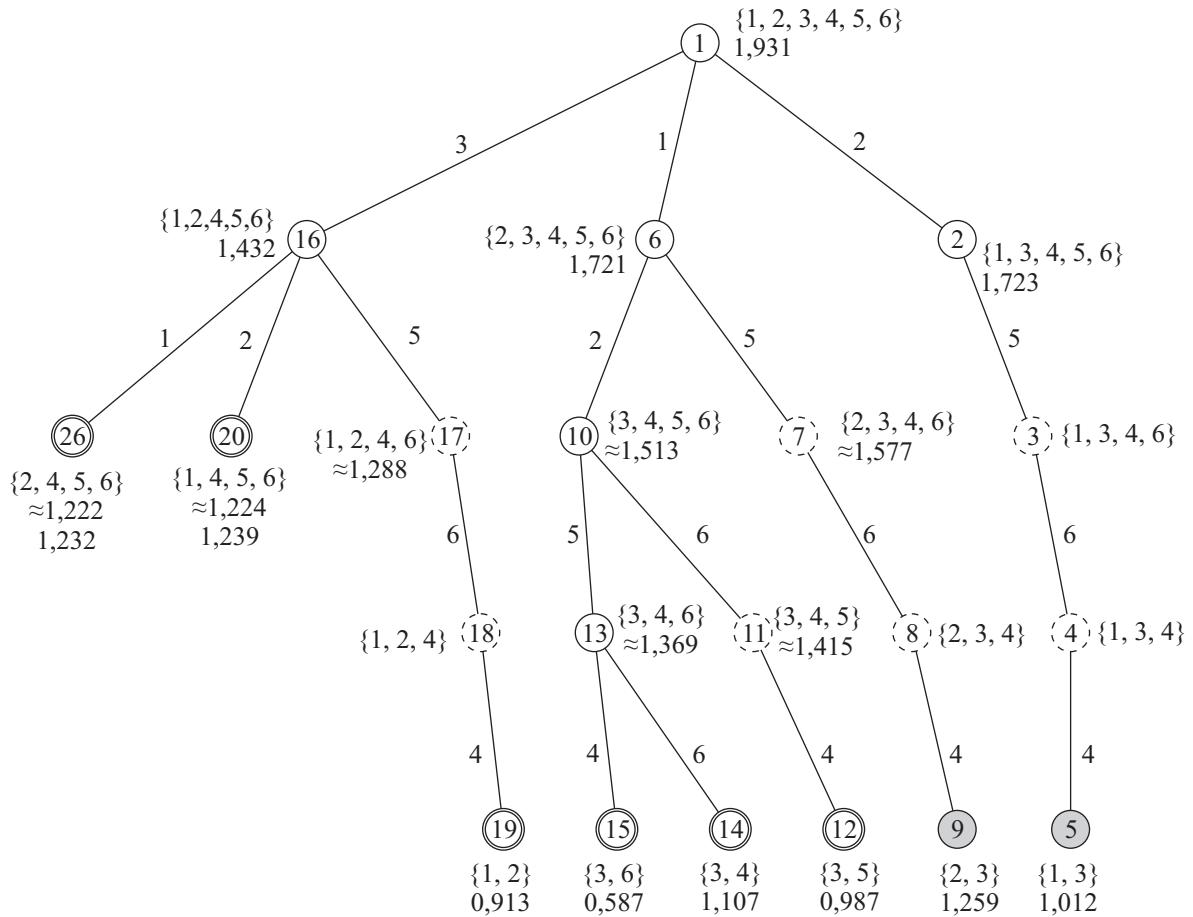


Figura 3.4: Árvore de busca do *branch and bound* rápido para  $D = 6$ ,  $d = 2$ ,  $\delta = 1$  e  $\gamma = 1$ . As notações utilizadas nessa figura são as mesmas da Figura 3.2. Os valores próximos aos nós acompanhados do símbolo  $\approx$  indicam a previsão  $\hat{J}(\cdot)$ . Quando um outro valor também está próximo ao nó, o valor real  $J(\cdot)$  também foi calculado, como nos nós 20 e 26. Diferentemente das Figuras 3.2 e 3.3, os nós eliminados pelas podas não são mostrados. Os nós com contorno tracejado foram omitidos do percurso por ser empregada a árvore de busca mínima. O nó 2 não foi omitido, apesar de possuir apenas um sucessor, pois o valor de  $J(\cdot)$  correspondente já foi calculado na etapa de ordenação.

é verificada novamente. Isso é necessário para garantir que a solução encontrada pelo algoritmo seja a ótima. Essa substituição ocorre nos nós 20 e 26 da Figura 3.4. Os valores padrões para os parâmetros são  $\delta = 1$  e  $\gamma = 1$  (Somol et al., 2004).

O ganho em eficiência é grande com o uso de previsões, pois é possível ordenar as características em cada expansão sem o aumento do número de chamadas de  $J(\cdot)$ . A maior parte das chamadas de  $J(\cdot)$  é realizada para atualizar o valor de  $B$  e em prováveis situações de podas.

### 3.2.6 *Branch and Bound* com Previsão Parcial

A árvore de busca mínima e as previsões do *branch and bound* rápido interrompem a seqüência de chamadas da função critério e impedem o cálculo recursivo. O *branch and bound* com previsão parcial é uma modificação do *branch and bound* rápido que realiza previsões apenas na etapa de ordenação e não utiliza a árvore de busca mínima. Com exceção da raiz, o valor de  $J(\cdot)$  é calculado recursivamente para todos os nós. Assim, as podas sempre são realizadas nos nós exatos, independente da qualidade das previsões. Apesar do cálculo recursivo ser mais rápido do que o não recursivo, o número de chamadas da função critério é maior nessa versão do algoritmo do que no *branch and bound* rápido. Em testes realizados, o *branch and bound* com previsão parcial não apresentou eficiência superior ao *branch and bound* rápido.

### 3.2.7 Busca da Direita para a Esquerda

Como apresentado na seção 3.2.1, as podas são possíveis na árvore do *branch and bound* pois a função critério é monotônica. Ou seja, se  $X_{k+i} \subset X_k$ , então  $J(X_{k+i}) \leq J(X_k)$ , para  $i > 0$ . Porém, da maneira como a árvore é construída, pode existir algum  $X_{k+i} \subset X_k$  em algum nó fora da subárvore iniciada por  $X_k$ . Assim,  $X_{k+i}$  não é eliminado quando ocorre a poda dessa subárvore. Um exemplo dessa situação pode ser visto na Figura 3.2: o subconjunto  $\{4, 5, 6\}$ , do nó 32, está contido em  $\{2, 4, 5, 6\}$ , do nó 20.

A busca da direita para a esquerda consiste no armazenamento dos subconjuntos quando o nó correspondente sofre poda. No percurso pela árvore, os novos subconjuntos analisados são comparados com os subconjuntos armazenados. Se um subconjunto estiver contido em outro, a poda é realizada sem a chamada da função critério. Apesar de essa estratégia reduzir o número de chamadas da função critério, a comparação de muitos subconjuntos é custosa computacionalmente (Nakariyakul e Casasent, 2007).

### 3.2.8 *Branch and Bound* Adaptativo

O *branch and bound* adaptativo é uma das versões mais eficientes desse método de seleção de características, especialmente quando  $D > 30$ . Diversas estratégias são utilizadas para

isso. A Figura 3.5 mostra um exemplo da árvore de busca desse algoritmo.

A ordenação é realizada apenas uma vez de acordo com a significância das características. A característica mais significativa de um conjunto é aquela que leva ao subconjunto com o menor valor da função critério quando removida. Uma explicação mais ampla sobre significância é feita na Seção 3.3. Inicialmente, a característica mais significativa do conjunto  $Y$  é selecionada e armazenada em  $y_1$ . Em seguida, a característica mais significativa do subconjunto  $Y \setminus \{y_1\}$  é selecionada e armazenada em  $y_2$ . O processo se repete até a ordenação de todas as características, resultando na seqüência ordenada  $\Omega = (y_1, y_2, \dots, y_D)$ . Essa ordenação coincide com a ordem em que as características são removidas no caminho do lado mais à esquerda da árvore do *branch and bound* ordenado.

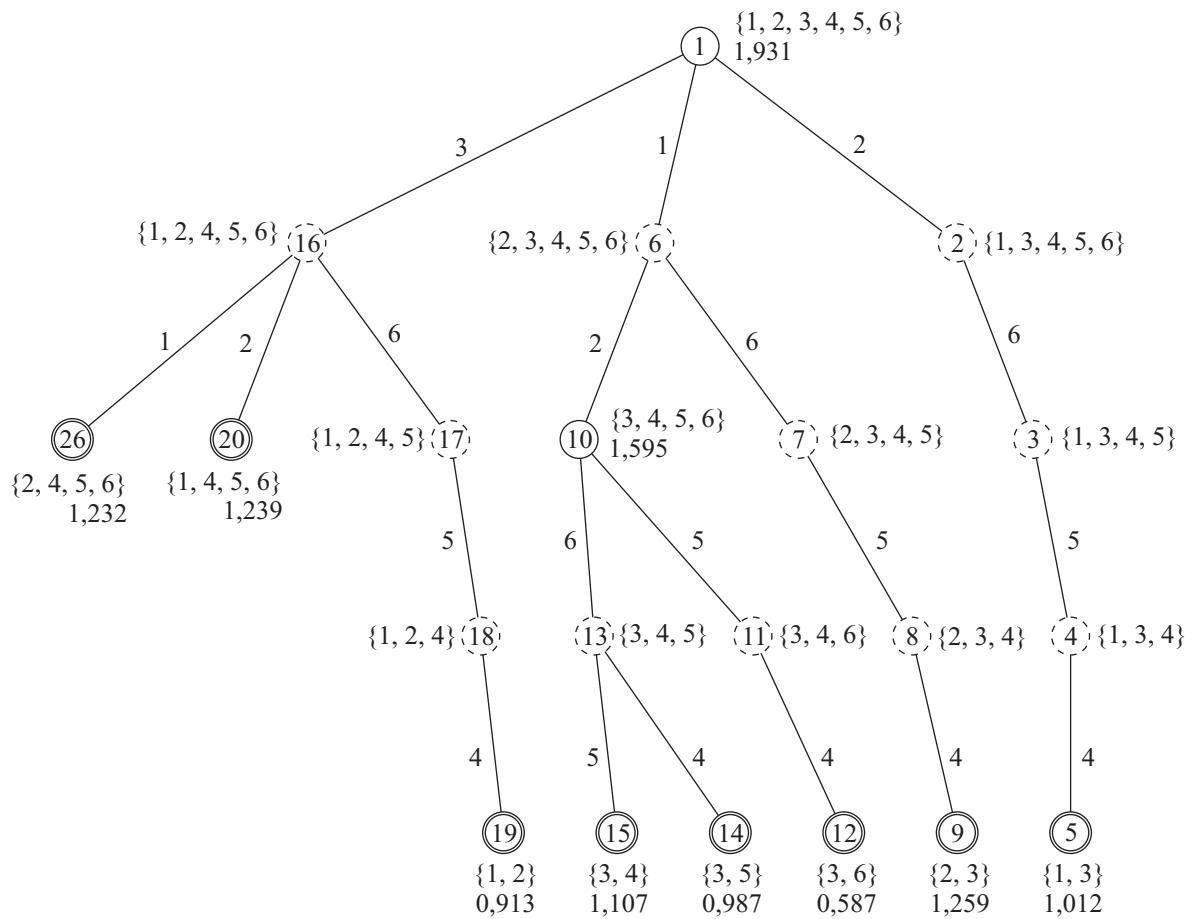


Figura 3.5: Árvore de busca do *branch and bound* adaptativo para  $D = 6$  e  $d = 2$ . As notações utilizadas nessa figura são as mesmas da Figura 3.2. A ordenação inicial das características é  $\Omega = (3, 1, 2, 6, 5, 4)$ . A solução inicial é  $X' = \{2, 3\}$  e o limite inicial é  $B' = J(X') = 1,259$ . O nível inicial de busca é  $k' = 2$ . Os nós 2, 6, e 16 foram omitidos por estarem em um nível menos do que  $k'$  e não serem a raiz. O nó 13 foi omitido devido à previsão realizada no nó 10. Os outros nós com contorno tracejado foram omitidos devido ao uso da árvore de busca mínima. O limite não foi atualizado em nenhuma folha pois a solução inicial já é ótima.

Quanto maior o valor de  $B$ , maior a probabilidade de ocorrerem podas em níveis mais próximos da raiz, o que leva a um número maior de eliminações. O conhecimento de um valor alto para  $B$  antes do início do percurso pela árvore contribui para isso. No *branch and bound* adaptativo, o limite inicial  $B'$  e o subconjunto correspondente são obtidos com o SFFS, já que esse algoritmo é capaz de achar um subconjunto ótimo ou próximo do ótimo rapidamente. A utilização de outro algoritmo de seleção para obtenção de um limite inicial também foi proposto anteriormente ao *branch and bound* adaptativo(Kudo e Sklansky, 2000).

O percurso do algoritmo pela árvore é realizado em “saltos” pelos níveis. Assim, o algoritmo calcula  $J(\cdot)$  apenas nos níveis em que as podas são mais prováveis. O nível inicial  $k'$  da busca é determinado com base na ordem das características e no valor de  $B'$  já obtidos:

$$k' = \operatorname{argmin}_{k=1,\dots,D} \left( \left\{ J(Y \setminus \{y_1, y_2, \dots, y_k\}) \mid J(Y \setminus \{y_1, y_2, \dots, y_k\}) < B' \right\} \right)$$

Os valores de  $J(\cdot)$  não são calculados em níveis menores do que  $k'$ , com exceção de  $J(Y)$ , necessário no mecanismo de previsão. No nível  $k'$  ocorrerá poda pelo menos do nó mais à esquerda da árvore, pois esse é o nó correspondente ao subconjunto  $Y \setminus \{y_1, y_2, \dots, y_{k'}\}$ . Essa etapa do processo não requer mais cálculos de  $J(\cdot)$ , pois tais valores já foram calculados na etapa de ordenação.

No percurso pela árvore, o valor de  $J(Y)$  é calculado no início. Posteriormente,  $J(\cdot)$  volta a ser calculado apenas no nível  $k'$ . Na expansão de um nó desse nível, caso não ocorra poda, é previsto o próximo nível em que podas serão prováveis, ou seja, é determinado o próximo nível para salto na subárvore cuja raiz é esse nó. A previsão é feita com base na equação

$$\hat{J}(X_k) = J(Y) \cdot (1 - (k/D)^\beta). \quad (3.14)$$

Quando  $k = 0$ , ou seja, nenhuma característica foi removida,  $\hat{J}(X_0) = J(Y)$ . Quando  $k = D$ , ou seja, todas as características foram removidas,  $\hat{J}(X_D) = \hat{J}(\{\}) = 0$ . Para outros valores de  $k$ ,  $\hat{J}(X_k)$  varia em função de  $\beta$ . Sendo  $X_k$  o subconjunto correspondente ao nó analisado, a previsão é realizada inicialmente com o cálculo de  $\beta$ , substituindo-se  $\hat{J}(X_k)$  por  $J(X_k)$  na Equação 3.14, o que resulta em

$$\beta = \frac{\log(1 - J(X_k)/J(Y))}{\log(k/D)}. \quad (3.15)$$

Partindo-se do princípio de que o valor de  $\beta$  é semelhante para todos os nós de um caminho

da raiz até uma folha, o próximo nível onde ocorrerá poda é previsto pela equação

$$k = \left\lceil D \cdot (1 - B/J(Y))^{1/\beta} \right\rceil, \quad (3.16)$$

obtida pela substituição de  $\hat{J}(X_k)$  por  $B$  na Equação 3.14. Assim, na subárvore do nó analisado,  $J(\cdot)$  não é calculado em níveis inferiores ao que foi previsto pela Equação 3.16. Essa regra não é seguida nas folhas, onde o valor de  $J(X_{D-d})$  é sempre calculado. Caso não ocorra poda, o processo de previsão é repetido e um novo nível para o salto é calculado. O mecanismo de previsão é utilizado no nó 10 da Figura 3.5. Os cálculos realizados nesse caso foram

$$\beta = \frac{\log(1 - 1,595/1,931)}{\log(2/6)} = 1,592, \quad (3.17)$$

$$k = \left\lceil 6 \cdot (1 - 1,259/1,931)^{1/1,592} \right\rceil = \left\lceil 3,091 \right\rceil = 4. \quad (3.18)$$

O *branch and bound* adaptativo também emprega a busca da direita para a esquerda e a árvore de busca mínima. O processo de ordenação inicial realiza  $(D \cdot (D+1))/2$  chamadas da função critério, o que é muito pouco perto do número total de chamadas realizadas no percurso pela árvore quando  $D$  é grande ( $D > 30$ ). A obtenção de um limite inicial com o SFFS realiza ainda menos chamadas a  $J(\cdot)$ . O mecanismo de previsão funciona bem para diferentes funções critério, sendo o principal responsável pela eficiência do algoritmo. A busca da direita para a esquerda reduz um pouco o número de chamadas de  $J(\cdot)$ , mas é um processo muito lento e acaba prejudicando o desempenho total do algoritmo. Em experimentos realizados, optou-se por não utilizar essa estratégia.

### 3.3 Busca Seqüencial

Uma maneira muito simples de seleção de características é avaliar independentemente todas as  $D$  características de  $Y$ , ranqueá-las e selecionar as  $d$  mais bem ranqueadas. Esse é um processo rápido mas que raramente leva a um bom resultado. Supondo-se a situação em que duas características separam completamente os exemplos de uma base de dados entre as classes. Independentemente, essas características podem separar mal e não seriam selecionadas por esse método. Portanto, as características devem ser avaliadas juntas para que o resultado seja aceitável.

Diversos métodos realizam adição e/ou remoção de características seqüencialmente a partir de um subconjunto inicial, com chamadas da função critério em cada etapa. Com isso, subconjuntos de características são avaliados e não características individualmente (com exceção das etapas em que os subconjuntos possuem uma característica). Essa abordagem é chamada nesta dissertação de **busca seqüencial**. Antes da explicação desses métodos, a definição de **significância** será apresentada em dois contextos: a

significância de uma característica de um subconjunto e em relação a um subconjunto.

Seja  $W_k = \{w_1, w_2, \dots, w_k\}$  um subconjunto de  $k$  características do conjunto  $Y = \{y_1, y_2, \dots, y_D\}$ . A característica  $w_i$  **mais significativa** (melhor) **do subconjunto**  $W_k$  é definida por

$$w_i = \operatorname{argmin}_{w_j \in W_k} (J(W_k \setminus \{w_j\})). \quad (3.19)$$

A característica  $w_i$  **menos significativa** (pior) **do subconjunto**  $W_k$  é definida por

$$w_i = \operatorname{argmax}_{w_j \in W_k} (J(W_k \setminus \{w_j\})). \quad (3.20)$$

A característica  $w_i$  **mais significativa** (melhor) **em relação ao subconjunto**  $W_k$  é definida por

$$w_i = \operatorname{argmax}_{w_j \in Y \setminus W_k} (J(W_k \cup \{w_j\})). \quad (3.21)$$

A característica  $w_i$  **menos significativa** (pior) **em relação ao subconjunto**  $W_k$  é definida por

$$w_i = \operatorname{argmin}_{w_j \in Y \setminus W_k} (J(W_k \cup \{w_j\})). \quad (3.22)$$

Os primeiros métodos propostos de busca seqüêncial foram o *sequential backward selection* (SBS) (Marill, 1963 apud Pudil et al., 1994) e o *sequential forward selection* (SFS) (Whitney, 1971 Pudil et al., 1994). O SFS parte do conjunto vazio e adiciona sucessivamente a característica mais significativa em relação ao subconjunto obtido na etapa anterior, isto é, a busca é realizada para frente. O SBS realiza o processo inverso, parte do conjunto  $Y$  e remove sucessivamente a características menos significativa em relação ao subconjunto obtido na etapa anterior, isto é, a busca é realizada para trás. A parada do algoritmo acontece quando o número desejado de características  $d$  foi alcançado ou quando todas as características foram adicionadas ou removidas. No segundo caso, as soluções para todos os tamanhos de subconjunto são obtidas. O SFS normalmente é usado quando se deseja selecionar poucas características e o SBS, quando se deseja selecionar muitas. Esses métodos são rápidos e simples. Porém, facilmente ficam presos em uma solução ótima local que não necessariamente é próxima da solução ótima global. Quando uma característica é adicionada (ou removida), tal situação permanecerá assim até o fim da busca, mesmo que o contexto mude e essa característica perca importância (ou ganhe importância).

Uma maneira de abordar esse problema é mudar a direção da busca na mesma execução, fazendo adições e remoções de características. O *plus-l take-away-r* (PTA( $l, r$ ) )

(Stearns, 1976 apud Pudil et al., 1994) realiza a adição de  $l$  características e, em seguida, a remoção de  $r$  características sucessivamente, tal que  $l \neq r$ . Se  $l > r$ , o conjunto inicial deve ser o conjunto vazio. Se  $l < r$ , o conjunto inicial deve ser o conjunto  $Y$ . O SFS é equivalente ao PTA(1, 0), enquanto que o SBS é equivalente ao PTA(0, 1). Esse algoritmo tem maior probabilidade de encontrar a solução ótima ou próxima da ótima em relação às versões mais simples de busca seqüencial. Entretanto, possuem dois parâmetros que precisam ser definidos pelo usuário. Os métodos *sequential floating forward selection* (SFFS) e *sequential floating backward selection* (SBFS) foram propostos para que a mudança da direção da busca fosse realizada automaticamente, sem uso de parâmetros (Pudil et al., 1994). Em cada iteração do SFFS, partindo do conjunto vazio, uma característica é adicionada e zero ou mais características são removidas enquanto forem encontrados subconjuntos melhores do que os obtidos até então. O SBFS é análogo, partindo do conjunto  $Y$  e realizando uma remoção e zero ou mais adições em cada iteração. Para melhorar o resultado e evitar a busca por todos os tamanhos de subconjunto, um intervalo  $\Delta$  pode ser usado na parada. Ou seja, a busca é finalizada quando é encontrado um subconjunto de  $d + \Delta$  características no caso do SFFS ou  $d - \Delta$  no caso do SBFS. A existência do  $\Delta$  não compromete tanto o fundamento do método, que é evitar a necessidade de parâmetros, pois esse valor influencia pouco o desempenho e o resultado do algoritmo.

A Figura 3.6 mostra o algoritmo do SFFS. Essa implementação foi baseada no código-fonte disponível no site da universidade\* de alguns dos autores do método. Observa-se que a busca para trás parte do último subconjunto obtido com a busca para frente, identificado por  $W'$ , mesmo que este não seja o melhor subconjunto obtido até então. Algumas mudanças são desejáveis para que o algoritmo seja usado na prática. Por exemplo, os valores da função critério calculados na linha 8 podem ser armazenados para reutilização na linha 10.

## 3.4 Redes Neurais Artificiais

Redes neurais artificiais são sistemas computacionais formados pela interconexão de elementos chamados nós ou neurônios. Valores numéricos são processados em conjunto pelos neurônios e a saída de uma função é obtida. Originalmente, o funcionamento foi inspirado no cérebro humano e posteriormente foram incluídos conceitos de estatística e processamento de sinais. Entre as principais propriedades estão a possibilidade de paralelismo e a capacidade de aprendizado por exemplos. As redes podem ser organizadas de muitas maneiras diferentes, variando-se a topologia e o algoritmo de treinamento, entre outros parâmetros. Existem diferentes modelos de rede, sendo que cada um é apropriado para determinados tipos de problemas.

Os neurônios naturais possuem três partes principais: o corpo celular, os dendritos

---

\*<http://ro.utia.cz/>

```

1: função SFFS( $D, d, \Delta$ )
2:    $Y \leftarrow \{1, 2, \dots, D\}$ 
3:   para  $i \leftarrow 0$  até  $D$  faça
4:      $W_i \leftarrow \{\}$ 
5:   fim para
6:    $k \leftarrow 0$ 
7:   enquanto  $(k < d + \Delta) \wedge (k < D)$  faça
8:      $w^+ \leftarrow \operatorname{argmax}_{w_j \in Y \setminus W_k} (J(W_k \cup \{w_j\}))$             $\triangleright$  Busca para frente.
9:      $W' \leftarrow W_k \cup \{w^+\}$ 
10:    se  $J(W') > J(W_{k+1})$  então
11:       $W_{k+1} \leftarrow W'$ 
12:    fim se
13:     $k \leftarrow k + 1$ 
14:    saiu_do_laço  $\leftarrow 0$ 
15:    enquanto  $(k > 1) \wedge (\text{saiu\_do\_laço} = 0)$  faça
16:       $w^- \leftarrow \operatorname{argmax}_{w_j \in W'} (J(W' \setminus \{w_j\}))$             $\triangleright$  Busca para trás.
17:       $W' \leftarrow W' \setminus \{w^-\}$ 
18:      se  $J(W') > J(W_{k-1})$  então
19:         $W_{k-1} \leftarrow W'$ 
20:         $k \leftarrow k - 1$ 
21:      senão
22:        saiu_do_laço  $\leftarrow 1$ 
23:      fim se
24:    fim enquanto
25:  fim enquanto
26:  retorna  $(W_d, J(W_d))$ 
27: fim função

```

Figura 3.6: Algoritmo do SFFS.

e o axônio. Os dendritos recebem impulsos nervosos e os conduz até o corpo celular, onde é realizado o processamento. Os impulsos resultantes são transmitidos aos neurônios seguintes pelo axônio. A conexão entre um dendrito e um axônio é chamada de sinapse.

Um neurônio artificial segue princípios parecidos com os dos neurônios naturais. A Figura 3.7 mostra um neurônio  $k$  do tipo McCulloch-Pitts (McCulloch e Pitts, 1943) recebendo uma entrada de  $m$  dimensões, ou seja, um vetor com  $m$  elementos. Um neurônio sozinho faz o papel de uma função que divide o espaço de  $m$  dimensões por um hiperplano com uma certa atenuação. Os terminais  $x_1, x_2, \dots, x_m$  representam os valores numéricos das entradas dos  $m$  dendritos. Os pesos  $w_{k1}, w_{k2}, \dots, w_{km}$  podem ter valores positivos ou negativos para simular sinapses excitatórias ou inibitórias. O termo de polarização (*bias*) comporta-se como o peso de uma entrada extra de valor 1. Sem o termo de polarização, o hiperplano gerado pelo neurônio passaria necessariamente pelo ponto de origem do espaço. Os valores de entrada são multiplicados pelo peso correspondente e somados no corpo do neurônio:

$$v_k = \sum_{j=0}^m (w_{kj} \cdot x_j). \quad (3.23)$$

A saída é o resultado do valor da soma aplicado na função de ativação:

$$y_k = \varphi(v_k). \quad (3.24)$$

Diversas funções de ativação podem ser utilizadas, sendo que cada uma é adequada para determinados modelos de rede. Por exemplo, a função degrau é definida por

$$\varphi(v) = \begin{cases} 1 & \text{se } v > 0 \\ 0 & \text{se } v \leq 0. \end{cases} \quad (3.25)$$

A função de ativação sigmoidal é a mais utilizada em redes neurais artificiais. Um exemplo desse tipo de função é a logística, definida por

$$\varphi(v) = \frac{1}{1 + \exp(-a \cdot v)}, \quad (3.26)$$

sendo que o parâmetro  $a$  determina a suavidade da curva. A função sigmoidal é limitada, monotônica e tem comportamento entre linear e não-linear. Além disso, tem derivada contínua, uma exigência para o uso do algoritmo de aprendizado *backpropagation*, apresentado a seguir.

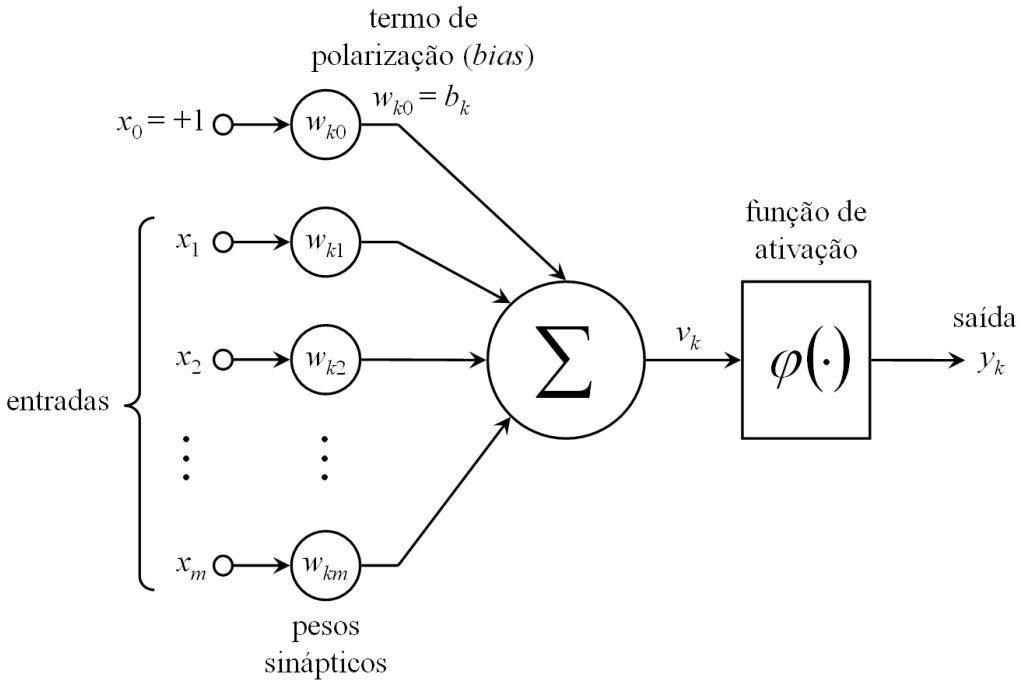


Figura 3.7: Modelo de um neurônio artificial (Haykin, 1999).

### 3.4.1 Multilayer Perceptron

Em redes neurais artificiais **acíclicas** (*feedforward* em inglês), os dados movem-se apenas na direção da entrada para a saída da rede. Nenhum neurônio pode receber como entrada a saída de um neurônio de uma camada seguinte. O **perceptron** com uma camada (Rosenblatt, 1958) é um tipo de rede neural acíclica e foi a primeira a ser desenvolvida. Sua arquitetura é formada apenas pela camada de entrada ligada aos neurônios da camada de saída. O treinamento é feito pelo algoritmo regra delta, em que uma função de erro é determinada pela diferença entre a saída da rede e a saída esperada e os pesos sinápticos são ajustados para minimização do erro.

Um perceptron com uma camada e um neurônio de saída é capaz de resolver apenas problemas linearmente separáveis. Com a adição de uma ou mais camadas intermediárias, também chamadas de camadas ocultas, constrói-se um **multilayer perceptron** (MLP, perceptron de múltiplas camadas em português), que permite a resolução de problemas não linearmente separáveis.

O número de camadas e número de neurônios em cada camada oculta normalmente é determinado empiricamente. Entre os fatores que influenciam esses parâmetros estão: o número de padrões de treinamento, a quantidade de ruído, a complexidade da função a ser aprendida e a distribuição estatística dos dados de treinamento (Braga et al., 2000). Camadas ocultas grandes podem fazer com que a rede memorize os padrões de

treinamento, enquanto que camadas ocultas pequenas podem levar à não convergência ou à maior generalização dos padrões de entrada.

Existem diversos algoritmos para treinamento de redes MLP, sendo que a maioria é uma modificação do *backpropagation* (Rumelhart e McClelland, 1986). Os algoritmos de treinamento estáticos alteram apenas os pesos sinápticos, enquanto que os dinâmicos podem alterar também o tamanho da rede.

O *backpropagation* é iterativo e dividido em duas etapas (Haykin, 1999). Primeiro é realizada a **propagação**: os valores de um padrão são apresentados à rede e processados, camada por camada, até a camada de saída, onde a resposta da rede é calculada. O erro da rede é determinado pela comparação da resposta obtida com a resposta desejada. Na segunda etapa, a **retropropagação**, o erro é transmitido da camada de saída até a primeira camada oculta e as variações dos pesos sinápticos são determinadas. Essas duas etapas são repetidas até que se satisfaça um critério de parada.

O treinamento consiste na apresentação dos padrões à rede e execução das duas etapas do *backpropagation*. Uma **época** é a apresentação do conjunto de treinamento inteiro. O treinamento pode ser realizado de maneira **seqüencial** (ou *on-line*), em que os pesos são atualizados quando cada padrão é apresentado. Pode ser vantajoso mudar a ordem dos padrões em cada época de maneira aleatória. Uma outra maneira é a **em lote**, em que a atualização dos pesos é realizada apenas uma vez por época.

Diversas abordagens podem ser utilizadas como critério de parada. O objetivo do treinamento é encontrar o vetor de pesos correspondente ao mínimo global da superfície de erro. É difícil saber se o ponto encontrado é um mínimo global ou local. Algumas abordagens são: parar quando a taxa de mudança do erro quadrático médio de uma época para outra ficar abaixo de um valor, parar depois de determinado número de épocas, parar quando o erro quadrático médio ficar abaixo de uma constante ou quando a taxa de padrões classificados corretamente ficar acima de uma constante.

### 3.4.2 Medida de Saliência

Se um algoritmo de empacotamento é empregado para seleção de características, a taxa de acerto de uma rede neural pode ser utilizada como função critério. Porém, o tempo de treinamento de uma rede neural é grande, o que inviabiliza o algoritmo na maioria dos casos. Uma alternativa é o cálculo da saliência de cada entrada de uma MLP após o treinamento (Garson, 1991; Nath et al., 1997; Castellano e Fanelli, 2000; Santos, 2007). Com isso, a relevância das características pode ser determinada sem repetidos treinamentos. A interpretação de pesos sinápticos, como é realizada na medição das saliências, é incomum, já que redes neurais normalmente são consideradas “caixas-pretas”, que simplesmente transformam uma entrada em uma saída. Originalmente, o método para o cálculo da saliência foi desenvolvido para ser aplicado em MLP’s com uma camada oculta e uma

saída (Garson, 1991; Nath et al., 1997). Uma extensão do método para diversas saídas (Santos, 2007) é apresentada a seguir.

A MLP analisada deve ser previamente treinada. Seja  $w_{ji}$  os pesos sinápticos da camada de entrada para a camada oculta e  $u_{kj}$  os pesos da camada oculta para a camada de saída. A rede possui  $p$  nós de entrada,  $q$  nós na camada oculta e  $h$  nós de saída. Portanto,  $i = 1, 2, \dots, p$ ,  $j = 1, 2, \dots, q$  e  $k = 1, 2, \dots, h$ . Os pesos  $w_{ji}$  são incorporados aos pesos  $u_{kj}$  por

$$w_{kji}^* = \frac{|w_{ji}| \cdot |u_{kj}|}{N_j}, \quad (3.27)$$

$$N_j = \sum_{i=1}^p w_{ji}. \quad (3.28)$$

A saliência da entrada  $i$  para a saída  $k$  é definida por

$$S_{ki} = \sum_{j=1}^q w_{kji}^*. \quad (3.29)$$

A saliência da entrada  $i$  em relação a todas as saídas é definida por

$$S'_i = \sum_{k=1}^h \frac{S_{ki}}{N'_k}, \quad (3.30)$$

$$N'_k = \sum_{i=1}^p S_{ki}. \quad (3.31)$$

### 3.5 Algoritmos Genéticos

Os algoritmos genéticos utilizam princípios naturais da evolução das espécies para resolver, em geral, problemas de otimização (Lacerda et al., 2002). Uma população inicial de cromossomos passa por uma seleção de acordo com a aptidão. Cada cromossomo representa uma possível solução. A aptidão é baseada na **função objetivo**, ou seja, a finalidade é encontrar o mínimo ou máximo global dessa função. Os cromossomos selecionados sofrem permutações e mutações, gerando uma nova população. O processo é repetido diversas vezes até a convergência para uma solução. Os cromossomos podem ser cadeias de valores binários ou reais, sendo que cada elemento é chamado de gene.

A **população inicial** deve apresentar uma boa dispersão pelo espaço de busca. Isso pode ser conseguido com um espaçamento uniforme entre cada cromossomo. No caso da representação binária, metade dos cromossomos pode ser gerada aleatoriamente e a outra metade pelo complemento dos cromossomos da primeira metade. Na técnica *seeding*, a população inicial é formada pelas soluções de outros algoritmos de otimização.

A **aptidão** pode ser igual à função objetivo. Porém, nesse caso, a aptidão pode

assumir valores negativos, o que é inadequado para alguns métodos de seleção. A alternativa pode ser a ordenação dos cromossomos de acordo com o valor da função objetivo e a atribuição de um valor com variação linear.

Cromossomos de diversas regiões do espaço de busca devem ser preservados para gerações seguintes. Eliminar os piores e manter apenas os melhores pode prender os cromossomos em um mínimo ou máximo local. A **seleção** pode ser feita com o algoritmo da roleta: os cromossomos são selecionados com probabilidade proporcional à sua aptidão. Uma variação desse algoritmo é a amostragem universal estocástica. O funcionamento pode ser comparado com o seguinte mecanismo: uma roleta é feita com um gráfico do tipo “torta”, sendo que a largura de cada fatia corresponde à aptidão de cada cromossomo. Depois de girar a roleta,  $N$  ponteiros colocados em volta do gráfico com espaçamento uniforme selecionam os cromossomos. Outro método é a seleção por torneio, em que são escolhidos  $n$  cromossomos e o de maior aptidão é escolhido. A escolha pode ser diferente: em vez de apenas o melhor, qualquer um pode ser escolhido, sendo que o primeiro tem probabilidade  $q$ , o segundo tem probabilidade  $q(q - 1)$ , o terceiro tem probabilidade  $q(q - 1)^2$  e assim por diante.

Alguns cromossomos da geração anterior podem ser intencionalmente mantidos na geração seguinte. Utilizando-se o elitismo, o de maior aptidão é preservado. Com a substituição de estado uniforme, apenas um ou dois cromossomos são substituídos em cada geração. Na substituição geracional, toda a população é substituída.

Para terminar de formar a geração seguinte, alguns cromossomos novos devem ser criados. Informações dos cromossomos já existentes são preservadas nos novos com a utilização de **permutações**. Ou seja, não são criados cromossomos aleatoriamente, mas, sim, pela combinação dos que foram selecionados. Isso pode ser feito pela divisão de dois cromossomos pais em  $n$  pontos, as partes são então trocadas, gerando dois cromossomos filhos. Ou pode ser utilizada uma máscara aleatória de bits que define qual gene dos pais vai para qual filho. Para aumentar a variabilidade genética é realizada a **mutação**, em que alguns genes escolhidos aleatoriamente são alterados. Normalmente a taxa de mutação é baixa.

Quando **números reais** são utilizados para compor os cromossomos, a combinação dos pais pode ser diferente. Na operação BLX- $\alpha$  ou permuta mista, o filho  $c$  é gerado por  $c = p_1 + \beta \cdot (p_2 - p_1)$ , sendo que  $p_1$  e  $p_2$  são os cromossomos pais,  $\beta$  é um valor aleatório com distribuição uniforme entre  $-\alpha$  e  $1 + \alpha$ . A direção da busca pode ser utilizada na geração dos filhos. Seja  $f(\cdot)$  a função objetivo que se deseja minimizar. O filho  $c$  é gerado por

$$c = \begin{cases} p_1 + r(p_1 - p_2) & \text{se } f(p_1) \leq f(p_2) \\ p_2 + r(p_2 - p_1) & \text{se } f(p_1) > f(p_2), \end{cases}$$

sendo que  $r$  é um valor aleatório com distribuição uniforme entre 0 e 1.

Na mutação de cromossomos com representação real, um gene pode ser escolhido aleatoriamente e seu valor alterado para qualquer um pertencente ao espaço de busca. Esse novo valor pode ser gerado com distribuição uniforme ou normal, tomando-se como média o valor anterior. Na mutação *creep*, a distribuição utilizada é normal com variância pequena para causar pequena perturbação no sistema.

Normalmente, o processo é realizado até que certa quantidade de cromossomos represente o mesmo valor, indicando convergência. Algoritmos genéticos exploram bem todo o espaço de busca, mas convergem lentamente quando a solução está próxima. Outro algoritmo de otimização pode ser utilizado no final do processo para acelerar a convergência.

### 3.5.1 Aplicação de Algoritmos Genéticos em Seleção de Características

Um algoritmo genético pode ser utilizado em seleção de características com o objetivo de minimizar o número de características e maximizar a taxa de acerto de um classificador (Siedlecki e Sklansky, 1989). Um processo baseado na proposta original é apresentado a seguir (a proposta original utiliza a taxa de erro no lugar da taxa de acerto).

Um cromossomo  $a = (\alpha_1, \dots, \alpha_n)$  representa um subconjunto de características, sendo que  $n$  é o número total de características e

$$\alpha_i = \begin{cases} 0 & \text{se a característica } i \text{ está presente} \\ 1 & \text{se a característica } i \text{ está ausente.} \end{cases}$$

O número de características do subconjunto,  $l(\cdot)$ , é calculado por:

$$l(a) = \sum_{i=1}^n \alpha_i \tag{3.32}$$

Uma penalidade  $p(\cdot)$  é calculada a partir da função critério  $J(\cdot)$ , que corresponde à taxa de acerto que o subconjunto obtém em um classificador. A penalidade é definida por

$$p(a) = \frac{\exp\left(\frac{t - J(a)}{m}\right) - 1}{\exp(1) - 1}, \tag{3.33}$$

sendo que  $t$  é um limiar de viabilidade e  $m$  é um fator de escala (margem de tolerância). O limiar de viabilidade deve ser definido como um valor mínimo aceitável para a taxa de acerto. Ou seja, espera-se encontrar um subconjunto de características que consiga obter uma taxa de acerto superior a  $t$ . A função  $p(a)$  tem valor negativo quando  $J(a) > t$ , valor

entre 0 e 1 quando  $t > J(a) > (t - m)$  e tem valor maior do que 1 quando  $J(a) < t - m$ . A função objetivo a ser minimizada é definida por

$$f(a) = l(a) + p(a). \quad (3.34)$$

Assim, subconjuntos de características com taxa de acerto superior ao limiar de viabilidade recebem uma pequena recompensa (penalidade negativa). Subconjuntos com o mesmo número de características são diferenciados pela taxa de acerto. Subconjuntos com taxa de acerto inferior a  $t$ , mas superior a  $t - m$ , recebem penalidade entre 0 e 1, o que permite que eles sejam melhores do que subconjuntos com uma característica a mais. O mesmo não ocorre com subconjuntos com taxa de acerto inferior a  $t - m$ , que dificilmente poderão competir com subconjuntos com uma característica a mais.

A taxa de permutação deve ter valores altos (0,8–0,6) e a taxa de mutação, valores baixos (0,4–0,01). Diferentes métodos de seleção e substituição de cromossomos podem ser utilizados.

## 3.6 Considerações Finais

A redução do número de características de uma base de dados é uma etapa importante do pré-processamento. Além de diminuir o custo computacional, essa redução pode aumentar a precisão no reconhecimento de padrões. Esse processo pode ser conseguido com a seleção de características. A seleção é vinculada à base de dados analisada, à função critério e ao objetivo (A, B ou C, como explicado no início deste capítulo). Portanto, a base de dados empregada na seleção deve representar bem o problema abordado. O uso de uma função critério adequada também é importante. Para garantir que a avaliação feita pela função critério seja eficiente, o próprio algoritmo de reconhecimento de padrões pode ser empregado, sendo assim uma função critério dependente. Outra alternativa é usar uma função critério independente, que usa critérios estatísticos para medir a separação dos exemplos entre as classes.

Existem diferentes algoritmos para a busca de um subconjunto de características de acordo com a função critério. Alguns garantem que a solução encontrada é ótima, mas exigem muito tempo de execução. A busca exaustiva e o *branch and bound* são algoritmos ótimos. Outros procuram encontrar a solução ótima, mas podem achá-la ou não. Alguns exemplos são redes neurais, algoritmos genéticos e SFFS.

No Capítulo 4, uma nova estratégia para o *branch and bound*, chamada floresta, é apresentada. No Capítulo 5, são apresentados experimentos que utilizam seleção de características em diferentes problemas. Os algoritmos empregados nos experimentos foram a busca exaustiva, diferentes versões do *branch and bound* e o SFFS. A busca exaustiva foi incluída para avaliar o quão perto o SFFS consegue chegar da solução ótima.

As diferentes versões do *branch and bound* foram testadas para avaliar a estratégia floresta e a viabilidade de um método ótimo com características de textura. O SFFS foi incluído por ser amplamente utilizado devido à eficiência. Outros métodos, como redes neurais e algoritmos genéticos, fazem parte de outros projetos do grupo de processamento de imagens do ICMC (Santos, 2007).

## Nova Estratégia para o *Branch and Bound*

---

---

Em geral, o valor de uma função critério monotônica tende a ser maior para subconjunto maiores. Assim, quanto mais próximo um nó está da raiz, maior a probabilidade do valor de  $J(\cdot)$  correspondente ser grande. Portanto, podas em níveis próximos a raiz são menos freqüentes. A não ocorrência dessas podas implica em grande parte das chamadas da função critério nas versões mais recentes do *branch and bound*. Como será explicado a seguir, é possível a construção de outras árvores de busca para avaliar parte das possíveis soluções. Assim, alguns subconjuntos são agrupados em nós mais próximos às folhas, o que aumenta a probabilidade de eliminação. Essa estratégia é chamada de **floresta**, pois utiliza mais de uma árvore.

A Árvore 1 da Figura 4.1 é uma árvore de busca mínima construída de acordo com as regras apresentadas nas Seções 3.2.1 e 3.2.4 para  $D = 6$  e  $d = 2$ . Os nós com contorno tracejado são omitidos do percurso por possuírem apenas um sucessor. Supondo-se que nenhuma poda ocorra, por exemplo, em níveis  $k < 3$ , as folhas 5, 9, 12, 19, 22 e 28 não são eliminadas. Os nós do nível  $k = 3$  que possuem apenas um sucessor fazem parte dos caminhos que levam a essas folhas. A remoção das características de  $\{5, 6\}$  ocorre nesses e somente nesses caminhos. Com isso, essas são todas as folhas correspondentes aos subconjuntos formados apenas pelos elementos de  $\{1, 2, 3, 4\}$ . A Árvore 2 da Figura 4.1 contém apenas esses subconjuntos. Nessa árvore, a probabilidade de eliminação dessas folhas é maior.

Generalizando, seja  $s_1$  um determinado nível de uma árvore e  $n_1$  o número de características avaliadas nessa árvore. No caso da árvore original,  $n_1 = D$ . Os nós do nível  $s_1$  que possuem apenas um sucessor estão nos caminhos em que as características de

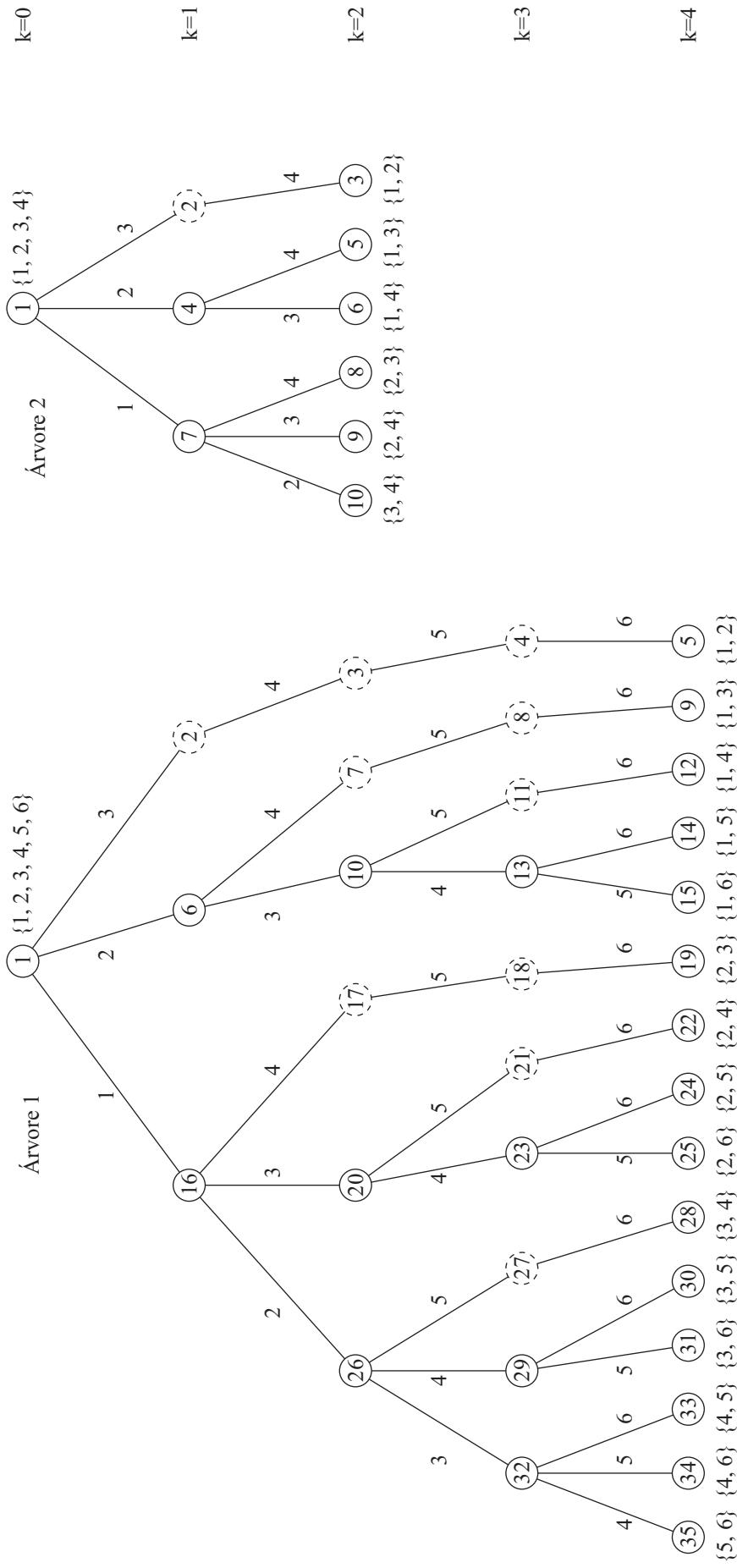


Figura 4.1: Floresta formada por duas árvores para  $D = 6$  e  $d = 2$ . A Árvore 2 contém os subconjuntos que seriam eliminados apenas por podas em níveis  $k < 3$  na Árvore 1 quando utilizada a busca mínima. De acordo com a notação utilizada,  $n_1 = 6$ ,  $n_2 = 4$  e  $s_1 = 3$ .

$\{d + s_1, d + s_1 + 1, \dots, n_1\}$  são removidas. A remoção de todas essas características não ocorre em nenhum outro caminho da raiz até uma folha. Essa afirmação é verdadeira, pois a remoção da característica  $d + s_1$  na passagem do nível  $s_1 - 1$  para o nível  $s_1$  implica em nós com apenas um sucessor do nível  $s_1$  em diante para que a Equação 3.7 seja válida. Assim, os caminhos que passam pelos nós do nível  $s_1$  com apenas um sucessor levam a todas as folhas correspondentes aos subconjuntos que contém apenas as características de  $\{1, 2, \dots, d + s_1 - 1\}$ .

Portanto, uma segunda árvore pode ser construída para avaliar as características de  $\{1, 2, \dots, n_2\}$ , sendo que  $n_2 = d + s_1 - 1$ . Essa segunda árvore deve ser construída com as mesmas regras da árvore original e o mesmo valor para  $d$ . Os subconjuntos correspondentes às folhas dessa árvore são exatamente os mesmos que seriam descartados apenas por podas em níveis  $k < s_1$  na árvore original. O mesmo processo pode ser utilizado para gerar uma terceira árvore a partir da segunda. De maneira geral, a seguinte equação é utilizada para se determinar o número de caracaterísticas avaliadas na  $i$ -ésima árvore:

$$n_i = \begin{cases} D & \text{se } i = 1 \\ d + s_{i-1} - 1 & \text{se } i > 1. \end{cases} \quad (4.1)$$

A ordenação das características não foi utilizada na explicação do método. Mas essa estratégia pode ser empregada sem comprometer a validade da geração das árvores. Para isso, a ordenação deve ser realizada apenas uma vez, antes do percurso por cada árvore, como no *branch and bound* adaptativo. No restante da explicação, as características serão representadas por variáveis, o que é suficiente para indicar a mudança na ordem das características. Assim, o conjunto de características avaliadas pela árvore  $i$  é a seqüência ordenada  $Z_i = (y_1, y_2, \dots, y_{n_i})$ .

É difícil saber quais são os níveis  $s_i$  ideais para que todas as árvores sejam criadas e o número de chamadas da função critério seja o menor possível. Será apresentada uma maneira de calcular  $s_i$  a partir do nível inicial de busca, que, por sua vez, é obtido de maneira similar à utilizada pelo *branch and bound* adaptativo.

Seja  $h_i = n_i - d$  o nível das folhas da árvore  $i$  e  $q_i$  o nível inicial de busca para a mesma árvore, sendo que

$$q_i = \underset{k=0,1,\dots,n_i}{\operatorname{argmin}} \left( \left\{ J(Z_i \setminus \{y_1, y_2, \dots, y_k\}) \mid J(Z_i \setminus \{y_1, y_2, \dots, y_k\}) < B' \right\} \right). \quad (4.2)$$

Se  $q_i = 0$ , a poda ocorrerá na raiz e nenhuma nova árvore deve ser gerada. Portanto,  $s_i = 0$ . Se  $q_i > 0$ , a ocorrência de podas em níveis inferiores a  $q_i$  é muito improvável.

Logo, é conveniente que  $q_i \leq s_i \leq h_i$ . O nível  $s_i$  é então determinado por

$$s_i = \begin{cases} 0 & \text{se } q_i = 0 \\ q_i + [\lambda(h_i - q_i)] & \text{se } 0 < q_i \leq h_i \\ h_i & \text{se } q_i > h_i \end{cases} \quad (4.3)$$

Sendo que  $0 \leq \lambda \leq 1$ . Ou seja, o parâmetro  $\lambda$  define uma posição entre  $h_i$  e  $q_i$ . Verificou-se empiricamente que um bom desempenho é obtido quando  $\lambda = 0,65$ .

Se utilizada a ordenação, as características mais significativas de uma árvore  $i$  são utilizadas para a construção da árvore  $i + 1$ . Portanto, a busca deve ser realizada da última para a primeira árvore gerada. Isso leva a avaliação dos melhores subconjuntos mais cedo. O percurso pela última árvore é realizado normalmente. Nas outras árvores, alguns nós são obsoletos, pois a poda dos mesmos eliminaria apenas folhas correspondentes a subconjuntos já avaliados. Assim, as podas não devem ser realizadas da maneira convencional, em que a função critério é chamada. Os nós obsoletos podem ser eliminados por podas automáticas dos nós que estão em um nível  $k \leq s_i$  e que possuem apenas um sucessor. Se  $s_i = h_i$ , as podas automáticas também devem ser realizadas nos sucessores mais à direita dos nós do nível  $k = h_i - 1$ . No exemplo da Figura 4.1, a poda automática deve ser realizada nos nós 2, 7, 11, 17, 21 e 27 da Árvore 1. A poda de nós que estão em níveis  $k < s_i$  e que possuem apenas dois sucessores levaria à remoção de nós obsoletos e de nós não obsoletos. Portanto, a poda automática não pode ser realizada em tais nós. O ideal é omitir esses nós da busca, ou seja, a função critério não deve ser chamada e a busca deve prosseguir. Na Figura 4.1, os nós 6, 10 e 20 da Árvore 1 devem ser omitidos.

## 4.1 Aplicação da Estratégia Floresta

A estratégia floresta é facilmente aplicada ao *branch and bound* adaptativo, que, por sua vez, já realiza algumas instruções utilizadas na estratégia proposta. As Figuras 4.2, 4.3 e 4.4 mostram o pseudocódigo do *branch and bound* adaptativo com a estratégia floresta. Inicialmente, devem ser realizados o cálculo de um limiar inicial com o SFFS (linha 2) e a ordenação (linha 4). A seguir, a função `gera_arvore()` é chamada para definir as especificações de cada árvore. O nível das folhas das árvores é calculado na linha 11, o nível inicial de busca é calculado na linha 12 e o número de características que serão avaliadas na árvore seguinte é calculado na linha 20. A função `expande()` é então chamada para a realização da busca. A verificação de poda automática, que elimina subconjuntos já avaliados em outras árvores, é feita na linha 28. Na linha 31, diversas verificações são realizadas para determinar se o subconjunto deve ser avaliado ou não. A avaliação deve ser feita se o nó for uma folha ( $k = h_i$ ) e não dever ser feita se o nível for menos do que `dest`, se o nó possuir apenas um sucessor ( $c = 1$ ) ou se o nó estiver em um nível inferior

```

1: função BB_adaptativo_floresta( $D, d, \lambda$ )
2:   Inicializar  $X'$  e  $B'$  com o SFFS.
3:    $B \leftarrow B'$ 
4:   Ordenar as características, resultando em  $\Omega = (y_1, y_2, \dots, y_D)$ .
5:    $n_1 \leftarrow D$ 
6:    $Z_1 \leftarrow \Omega$ 
7:   gera_arvore(1)
8:   retorna ( $X', B$ )
9: fim função

```

Figura 4.2: Função principal da estratégia floresta aplicada ao *branch and bound* adaptativo.

```

10: função gera_arvore( $i$ )
11:    $h_i \leftarrow n_i - d$ 
12:    $q_i \leftarrow \underset{k=0,1,\dots,n_i}{\operatorname{argmin}} \left( \left\{ J(Z_i \setminus \{y_1, y_2, \dots, y_k\}) \mid J(Z_i \setminus \{y_1, y_2, \dots, y_k\}) < B' \right\} \right)$ 
13:   se  $q_i = 0$  então
14:      $s_i \leftarrow 0$ 
15:   senão se  $0 < q_i \leq h_i$  então
16:      $s_i \leftarrow q_i + \lfloor \lambda(h_i - q_i) \rfloor$ 
17:   senão
18:      $s_i \leftarrow h_i$ 
19:   fim se
20:    $n_{i+1} \leftarrow d + s_i - 1$ 
21:    $Z_{i+1} \leftarrow (y_1, y_2, \dots, y_{n_{i+1}})$ 
22:   se  $s_i > 0$  então
23:     gera_arvore( $i + 1$ )
24:   fim se
25:   expande( $i, 0, Z_i, 0, d + 1, q_i$ )
26: fim função

```

Figura 4.3: Função gera\_arvore( $\cdot$ ).

```

27: função expande( $i, k, X, r, c, \text{dest}$ )
     $i$  é o número da árvore.
     $k$  é nível do nó atual.
     $X$  é o subconjunto do nó atual.
     $r$  é o índice da característica removida anteriormente.
     $c$  é o número de sucessores do nó atual.
     $\text{dest}$  é o destino previsto como sendo o nível da próxima poda.

28:   se ( $c = 1$ )  $\wedge$  ( $k \leq s_i$ ) então
29:     retorna
30:   fim se
31:   se ( $k = h_i$ )  $\vee$   $\neg((k < \text{dest}) \vee (c = 1) \vee (c = 2 \wedge k < s_i))$  então
32:     se  $J(X) \leq B$  então
33:       retorna
34:     fim se
35:     se  $k = h_i$  então
36:        $B \leftarrow J(X)$ 
37:        $X' \leftarrow X$ 
38:     retorna
39:   fim se
40:    $\beta \leftarrow \frac{\log(1 - J(X)/J(Z_i))}{\log(k/n_i)}$ 
41:    $\text{dest} \leftarrow \lceil n_i \cdot (1 - B/J(Z_i))^{1/\beta} \rceil$ 
42:   fim se
43:   para  $p \leftarrow c$  até 1 faça
44:     expande( $i, k + 1, X \setminus \{y_{r-1}\}, r - i, c - p + 1, \text{dest}$ )
45:   fim para
46: fim função

```

Figura 4.4: Função expande( $\cdot$ )

a  $s_i$  e possuir dois sucessores ( $c = 2 \wedge k < s_i$ ). Na avaliação, o valor de  $J(\cdot)$  é comparado com  $B$ , o que determina se deve ocorrer poda ou não (linha 32). Se chegar em uma folha, o limite  $B$  e o subconjunto  $X'$  são atualizados (linhas 36 e 37). Caso não ocorra poda na avaliação, um novo destino para poda é previsto (linhas 40 e 41). As chamadas da função para expansão dos nós sucessores são feitas no laço da linha 43.

A estratégia floresta também pode ser aplicada em outras versões do *branch and bound*. Porém, pode ser necessária alguma adaptação do algoritmo. Por exemplo, para o uso da estratégia com o *branch and bound* rápido, a reordenação durante o percurso pela árvore não pode ser feita. Em experimentos realizados, o algoritmo que apresentou melhor desempenho foi o *branch and bound* adaptativo com a floresta. Portanto, apenas essa versão foi apresentada nesta dissertação.

## 4.2 Comparação e Avaliação

Foi realizado um experimento utilizando a base de dados WDBC, obtida no repositório da UCI\*. Essa base de dados contém informações a respeito de células humanas e foi gerada para estudos a respeito de câncer de mama. A base contém 569 exemplos, sendo que 357 correspondem a amostras de tecidos benignos e 212, a amostras de tecido maligno. Cada exemplo possui 30 características contínuas. Como a base possui duas classes, a distância de Bhattacharyya foi empregada como função critério. Foram selecionados subconjuntos de características de tamanhos  $d = 1, 2, \dots, 29$  com diferentes versões do *branch and bound* e a busca exaustiva. Os tempos de processamento de cada execução são apresentados nos gráficos das Figuras 4.5 e 4.6. As especificações do computador utilizado podem ser vistas no Capítulo 5.

A busca exaustiva apresentou melhor desempenho para  $d < 4$ , enquanto que o

---

\*<http://archive.ics.uci.edu/ml/>

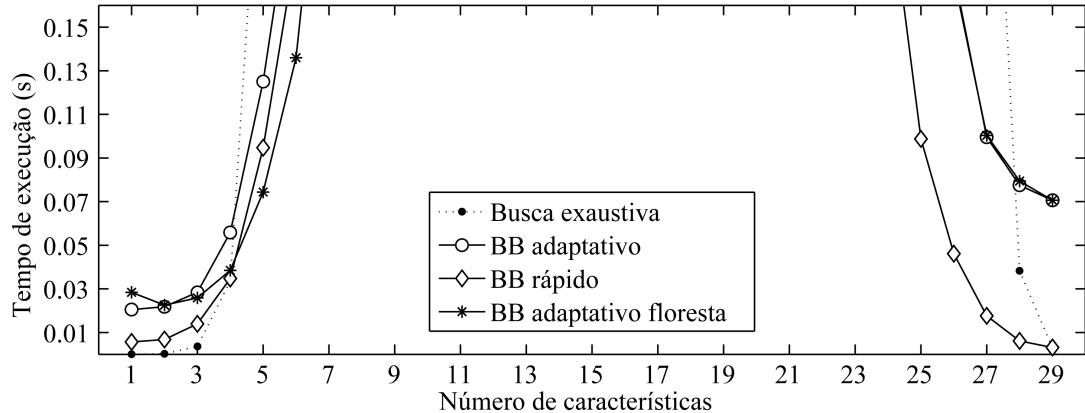


Figura 4.5: Tempo de execução de diferentes versões do *branch and bound* com maior ampliação.

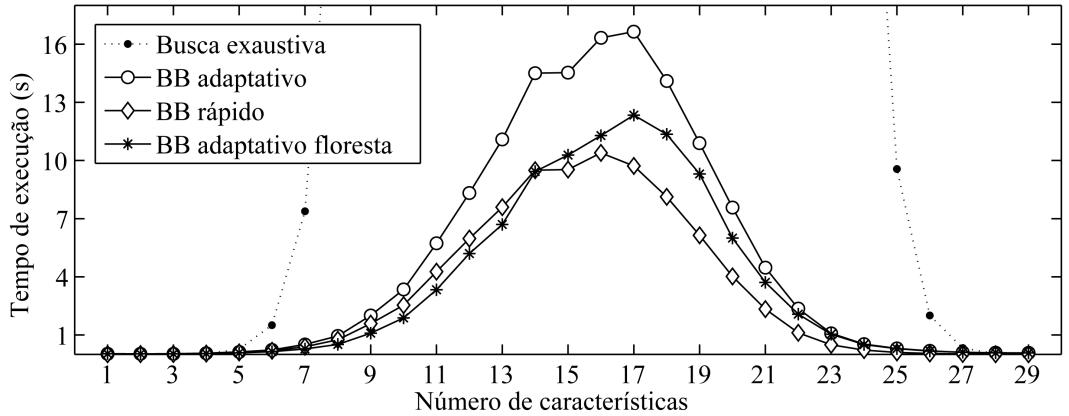


Figura 4.6: Tempo de execução de diferentes versões do *branch and bound* com menor ampliação.

*branch and bound* adaptativo com e sem a estratégia floresta apresentaram desempenho inferior. A causa para a lentidão do *branch and bound* adaptativo quando poucas características são selecionadas é a ordenação das características. Essa etapa do algoritmo exige certo tempo de processamento, mas é constante para qualquer número de características que se deseja selecionar. Como pode ser observado nos gráficos, esse tempo é pouco representativo quando  $d > 4$ . Fica evidente também que a busca exaustiva torna-se muito mais lenta do que o *branch and bound* para  $d > 5$ . A estratégia floresta tornou o algoritmo mais rápido do que os outros métodos quando  $5 \leq d \leq 14$ . As situações em que o número de características selecionadas é muito próximo do número total de características não são relevantes, pois não representam uma significante redução de dimensionalidade.

## Experimentos e Resultados

---

---

Os experimentos foram realizados para avaliar a seleção de características em diferentes problemas. Todos os problemas são relacionados ao uso de texturas em reconhecimento de padrões. Assim, os objetivos foram: (a) mostrar como a seleção de características melhora o reconhecimento de padrões; (b) mostrar que determinadas características são mais adequadas a cada tipo de problema e de imagem; (c) verificar que o uso de diferentes métodos de extração de características de texturas é melhor do que o uso de métodos isolados; e (d) comparar diferentes métodos de seleção e diferentes funções critério.

O primeiro experimento foi a classificação de regiões de uma foto aérea. As regiões deveriam ser classificadas como plantação de eucalipto, mata ou campo. As funções critério utilizadas foram a distância de JM e a taxa de acerto do CDM estimada com o *10-fold cross-validation* sem repetições ( $n = 1$ ) e com 80 repetições ( $n = 80$ ). Os métodos de seleção utilizados foram a busca exaustiva e o SFFS. Na validação, foi empregado o CDM.

O segundo experimento foi a segmentação não-supervisionada de mosaicos de texturas de Brodatz. As funções critério utilizadas foram distância de JM e a taxa de acerto do CDM estimada com o *10-fold cross-validation* sem repetições. Os métodos de seleção utilizados foram a busca exaustiva, diferentes versões do *branch and bound* e o SFFS. A validação, isto é, a segmentação não-supervisionada, foi realizada com o *k-means* (Banks, 1990). Nessa etapa, o comando `kmeans` do Matlab com os parâmetros padrões foi utilizado.

No terceiro experimento foi realizada a segmentação supervisionada de imagens de resonância magnética de cabeça. De 38 imagens, uma foi usada para treino e seleção de características e a demais, para validação. A seleção foi feita apenas com a taxa de acerto do CDM estimada com o *10-fold cross-validation* sem repetições e o SFFS. A validação

foi feita com o CDM.

Em relação a todos os experimentos, foram utilizadas características de matrizes de co-ocorrência, de Gabor e de estatísticas de primeira ordem. A redução do número de cores das imagens foi realizada com o comando `ditherr` do Matlab e parâmetros padrões. A distância euclidiana normalizada foi empregada no CDM, tanto na seleção de características como na validação. Na etapa de validação do segundo experimento, a base de dados foi normalizada antes do uso do *k-means*. Foi utilizado  $\Delta = 7$  para o SFFS. Os algoritmos de seleção de características, bem como as funções critério, foram implementados em C++. A execução da etapa de seleção foi realizada em um computador Sun V40z com quatro processadores Opteron 64 Dual Core de 2,2GHz modelo 875, 24GB de RAM e sistema operacional Solaris 2.10.

A divisão das bases de dados em partições foi realizada por um processo pseudo-aleatório, o que permite o uso de uma semente para que as mesmas partições sejam utilizadas em ocasiões diferentes. Isso foi aproveitado na função critério, para que a estimativa da taxa de acerto do CDM fosse equivalente para todos os subconjuntos de características e em diferentes algoritmos de seleção.

## 5.1 Classificação de Foto Aérea

As imagens utilizadas nesse experimento são regiões de uma foto aérea de plantação de eucalipto. A foto original é colorida e possui  $10264 \times 14276$  píxeis. Inicialmente a foto foi convertida para ficar com 256 tons de cinza distintos. Para que os diferentes tipos de terreno fossem incluídos nas regiões analisadas, definiu-se uma grade arbitrária  $15 \times 15$ , dividindo a imagem em células com aproximadamente  $684 \times 952$  píxeis. A Figura 5.1 mostra a foto aérea e a grade utilizada. Do canto superior esquerdo de cada célula, extraiu-se uma única região de interesse com  $64 \times 64$  píxeis. Seis dessas regiões são apresentadas na Figura 5.2. As regiões de interesse que continham alguma parte do fundo (cor branca) foram eliminadas, totalizando 130 imagens. Cada imagem foi classificada manualmente de acordo com o tipo de terreno predominante: (1) plantação de eucalipto, (2) mata e (3) campo. As classes estão indicadas ao lado das imagens da Figura 5.2. Ao todo, 61 imagens são de plantações de eucaliptos, 40 de matas e 29 de campos.

Foram extraídas 82 características de texturas de cada imagem, sendo que 36 foram obtidas com matrizes de co-ocorrência, 40 com filtros de Gabor e 6 com estatísticas de primeira ordem. As Tabelas Apêndice A.1, Apêndice A.2 e Apêndice A.3 no Apêndice Apêndice A mostraram os parâmetros empregados na extração de cada característica. O número de cores das imagens foi reduzido para 32 para a geração das matrizes de co-ocorrência. Para os demais métodos, o número de cores da imagem original foi preservado. Foram geradas 6 matrizes para cada imagem e as 6 funções apresentadas na Seção 2.1.2 foram aplicadas em cada matriz. Os filtros de Gabor foram gerados utilizando-se  $S = 20$ ,

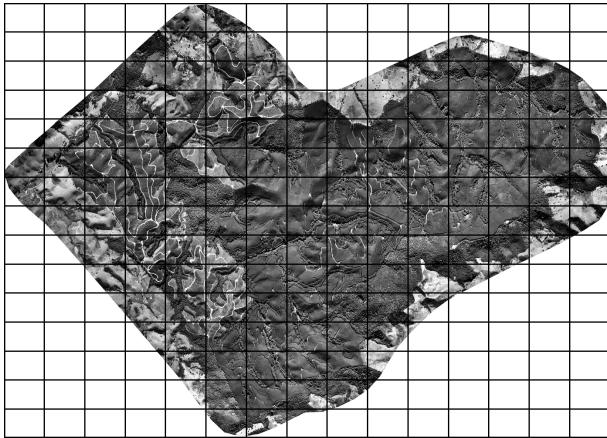


Figura 5.1: Foto aérea dividida em 15 linhas e 15 colunas.

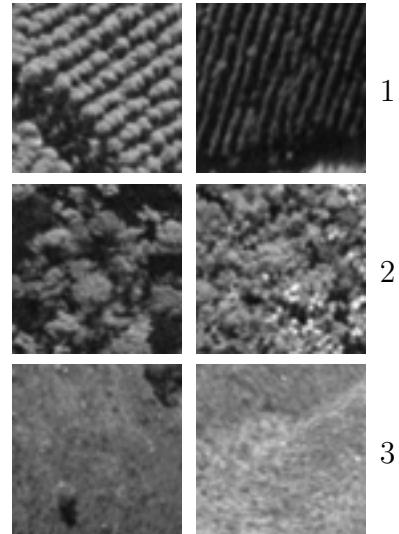


Figura 5.2: Exemplos das imagens analisadas. Os números à direita indicam as classes.

$K = 2$ ,  $U_l = 0,02$  e  $U_h = 0,3$ . Cada característica corresponde à energia (soma dos quadrados) da magnitude da imagem filtrada. As características de primeira ordem foram extraídas aplicando-se as 6 funções apresentadas na Seção 2.1.1 em cada imagem. Os parâmetros utilizados para extração das características com matrizes de co-ocorrência e filtros de Gabor foram determinados empiricamente. Assim, obteve-se uma base de dados com 130 exemplos, 82 características contínuas e 3 classes.

Diferentes métodos de seleção de características foram utilizados com o propósito de melhorar a precisão de um classificador de distância mínima. Assumindo-se que a seleção de características é uma etapa do processo de treino do classificador, o conjunto de treino deve ser utilizado na seleção. Para simplificar o experimento, optou-se por utilizar o método *holdout* para dividir da base de dados e validação dos resultados. Apesar do *holdout* não permitir uma estimativa muito exata da precisão do classificador, o método foi suficiente para mostrar os resultados da seleção de características. A base de dados foi dividida aleatoriamente em duas partições mutuamente exclusivas: o conjunto de treino, com 85 exemplos, e o conjunto de teste, com 45 exemplos.

A distância de JM foi utilizada como função critério independente. A distância de Bhattacharyya não pode ser utilizada isoladamente por haver mais de duas classes na base de dados. Como função critério dependente, foi utilizada a taxa de acerto do CDM. Verificou-se empiricamente que o classificador atinge uma precisão satisfatória se a distância euclidiana normalizada é empregada. Portanto, essa distância foi utilizada tanto na seleção de características como na classificação do conjunto de teste. A estimativa da taxa de acerto na seleção foi feita com o método *10-fold cross-validation* sem repetições ( $n = 1$ ) e com 80 repetições ( $n = 80$ ). O *10-fold cross-validation* sem repetições foi utilizado por

ser rápido. Porém, a seleção é comprometida pela baixa exatidão do método. O *10-fold cross-validation* com 80 repetições é mais exato para estimar a precisão do classificador, contudo, é mais lento. Em resumo, três funções critério foram avaliadas: distância de JM, taxa de acerto do CDM com  $n = 1$  e  $n = 80$ .

O BB não pode ser empregado para seleção com nenhuma das funções critério utilizadas. Essa restrição existe pois a taxa de acerto do CDM não é monotônica e a distância de JM inclui o cálculo do determinante e a inversão da matriz de covariância. Para a inversão e o cálculo do determinante, é necessário que existam mais exemplos do que características para cada classe na base de dados. Além disso, um número pequeno de exemplos aumenta a possibilidade de haver correlação entre características, o que também impede tais operações. O BB calcula o valor da função critério para subconjuntos grandes de características, impossibilitando o cálculo da distância de JM no caso desse experimento. Portanto, os algoritmos utilizados para seleção foram o SFFS e a busca exaustiva.

A distância de JM foi utilizada com a busca exaustiva para  $d = 1, 2, \dots, 6$  e com o SFFS para  $d = 2, 3, \dots, 22$ . A Figura 5.3 mostra o gráfico gerado com o valor da função critério correspondente a cada subconjunto selecionado. Observa-se que o SFFS conseguiu encontrar os subconjuntos ótimos em todos os casos que puderam ser verificados. Isto é, os valores da função critério para  $d = 2, 3, \dots, 6$  obtidos com o SFFS foram os mesmos obtidos pela busca exaustiva, que é um método ótimo. A busca exaustiva não foi utilizada para  $d > 6$  devido ao tempo de processamento, que seria muito grande. A seleção para  $d = 1$  não foi realizada com o SFFS pois assume-se que o melhor algoritmo nesse caso é sempre a busca exaustiva. A seleção para  $d > 22$  não foi possível devido às restrições para o cálculo da distância de JM (o conjunto de treino possui apenas 23 exemplos da classe 2). Observa-se também que o valor da função critério converge para 2,0. Se as características tivessem uma distribuição normal perfeita, isso indicaria uma separação completa entre as classes para valores muito próximos a 2,0.

A taxa de acerto do CDM com  $n = 1$  também foi utilizada com a busca exaustiva e o SFFS. A Figura 5.4 apresenta o gráfico gerado com o valor da função critério correspondente a cada subconjunto selecionado. A seleção com a busca exaustiva foi feita para  $d = 1, 2, \dots, 6$  e com o SFFS foi feita para  $d = 2, 3, \dots, 29, 30, 35, \dots, 75$ . Como o intuito da seleção de características é reduzir a dimensionalidade, mantendo-se poucas características, não houve a preocupação de um detalhamento grande na seleção de muitas características. Por isso nem todos os tamanhos de subconjuntos foram analisados entre 30 e 75. O gráfico da Figura 5.4 mostra apenas parte dos resultados obtidos. O SFFS não encontrou os subconjuntos ótimos em nenhum dos casos verificados. Mas chegou a soluções relativamente próximas das ótimas.

A outra função critério dependente, a taxa de acerto do CDM com  $n = 80$ , foi utilizada com os mesmos algoritmos de seleção das outras funções critério. A Figura 5.5

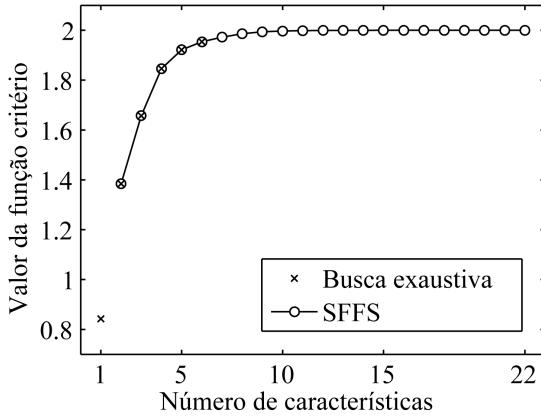


Figura 5.3: Distância de Jeffries-Matusita em relação ao número de características selecionadas.

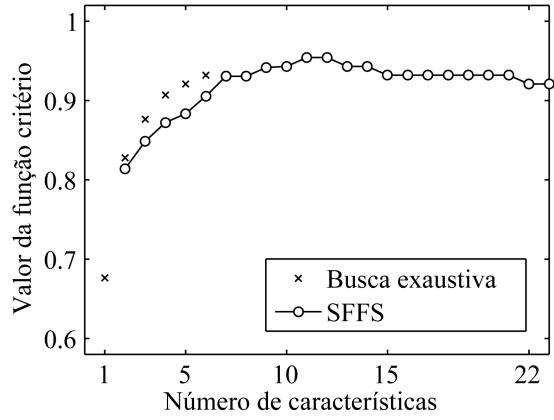


Figura 5.4: Taxa de acerto média do CDM  $n = 1$  em relação ao número de características selecionadas.

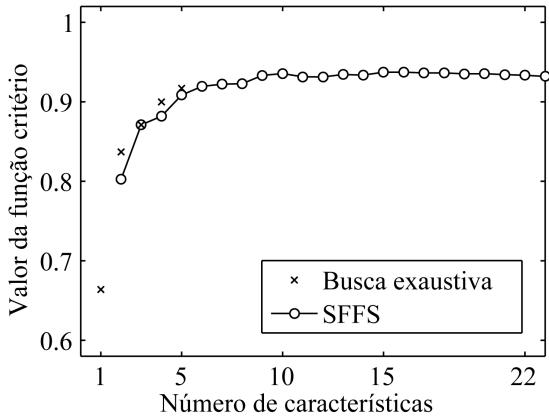


Figura 5.5: Valor da função critério CDM  $n = 80$  em relação ao número de características selecionadas.

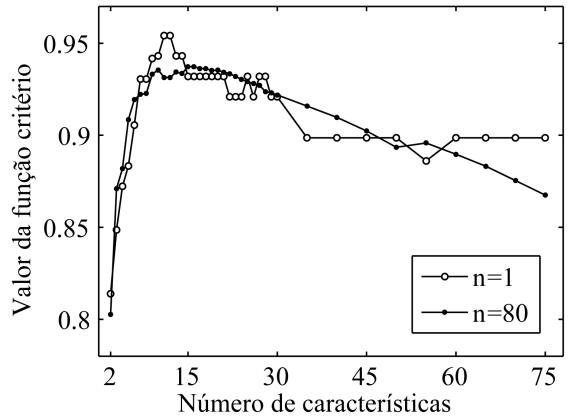


Figura 5.6: Valor da função critério CDM  $n = 1$  e CDM  $n = 80$  em relação ao número de características selecionadas pelo SFFS.

mostra o gráfico gerado com o valor da função critério correspondente a cada subconjunto selecionado. Para essa função critério, cujo cálculo é mais lento, a seleção com busca exaustiva foi utilizada para  $d = 1, 2, \dots, 5$ . Com o SFFS, os subconjuntos selecionados foram dos mesmos tamanhos abordados quando o CDM com  $n = 1$  foi utilizado. Apenas parte dos resultados são apresentados na Figura 5.5. Observa-se no gráfico que o SFFS obteve subconjuntos mais próximos dos ótimos com o CDM com  $n = 80$  do que com a função critério anterior. Para  $d = 3$ , a solução ótima foi encontrada. Possivelmente isso é conseqüência da maior exatidão do CDM com  $n = 80$  em estimar a taxa de acerto, se comparado com o CDM com  $n = 1$ .

Os resultados completos dos experimentos com o CDM com  $n = 1$  e  $n = 80$  são apresentados na Figura 5.6. Observa-se que existe um pico quando  $d = 11$  e  $d = 12$  para o

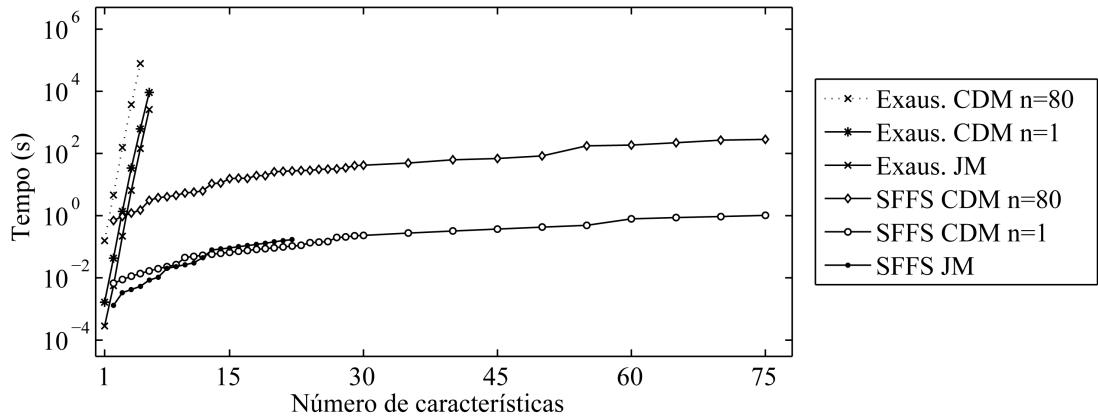


Figura 5.7: Gráfico semilog do tempo de execução da seleção em relação ao número de características selecionadas.

	1	2	4	5	6	75
Exaus. CDM $n = 80$	0,15s	4,57s	3.741,6s = 1,04h	77.868s = 21,63h	—	—
Exaus. CDM $n = 1$	0,0017s	0,043s	33,91s	617,09s	9146,9s = 2,54h	—
Exaus. JM	0,0003s	0,0056s	6,45s	144,54s	2.595,8s	—
SFFS CDM $n = 80$	—	0,69s	1,21s	1,51s	3,10s	285,27s
SFFS CDM $n = 1$	—	0,0067s	0,012s	0,013s	0,016s	1,03s
SFFS JM	—	0,0013s	0,0042s	0,0054s	0,0084s	—

Tabela 5.1: Tempo de execução da seleção de características.

CDM com  $n = 1$  e próximo a  $d = 16$  para o CDM com  $n = 80$ . Isso mostra uma provável melhoria com a seleção de características para o problema analisado. É possível notar também que a variação do valor da função critério em relação a  $d$  é mais suave quando utilizado o CDM com  $n = 80$ . As duas curvas são próximas, já que se trata da mesma base de dados e do mesmo classificador. Porém, a menor exatidão do CDM com  $n = 1$  contribui para uma maior instabilidade nos resultados.

O tempo de execução variou consideravelmente entre os métodos de seleção. Portanto, foi necessário o uso de um gráfico semilog para que os resultados pudessem ser visualizados. A Figura 5.7 mostra o gráfico e a Tabela 5.1 mostra alguns dos tempos. O tempo de execução do SFFS cresce pouco em relação ao número de características selecionadas se comparado com a busca exaustiva. O CDM com  $n = 80$ , mesmo sendo mais lento do que as outras funções critério, é viável computacionalmente quando utilizado com o SFFS.

O resultado da seleção de características é mais bem avaliado quando um conjunto de exemplos não empregado na seleção é utilizado. Todos os subconjuntos selecionados pelos diversos métodos foram utilizados no processo de validação. O CDM com distância euclidiana normalizada foi empregado para classificar os 45 exemplos do conjunto de

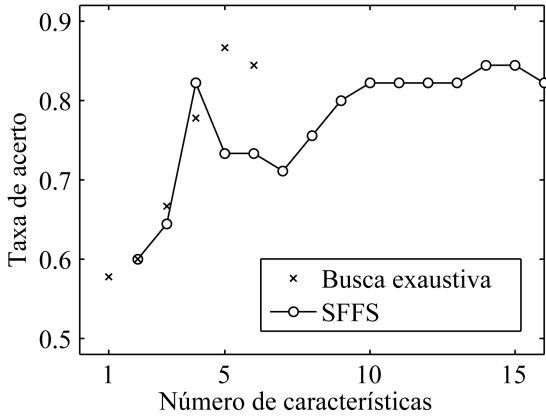


Figura 5.8: Taxa de acerto obtida na validação em relação ao número de características selecionadas com o CDM  $n = 1$ .

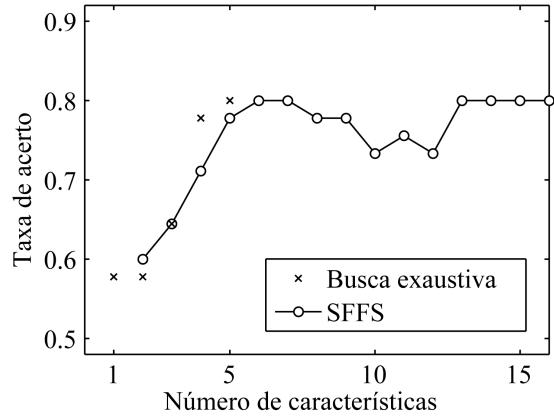


Figura 5.9: Taxa de acerto obtida na validação em relação ao número de características selecionadas com o CDM  $n = 80$ .

teste e as taxas de acerto foram calculadas. A comparação dos resultados obtidos com os subconjuntos selecionados com o CDM com  $n = 1$  e  $n = 80$  pode ser vista nas Figuras 5.8 e 5.9, respectivamente. Nota-se que há casos em que a taxa de acerto de subconjuntos ótimos (encontrados com a busca exaustiva) foi inferior à obtida com subconjuntos selecionados pelo SFFS. Isso não significa que o SFFS é melhor do que a busca exaustiva. O propósito do SFFS é sempre tentar encontrar ou chegar próximo da solução ótima. Uma possível explicação para esse comportamento é a falta de exatidão do *holdout*. Além disso, a seleção pode ficar muito específica para o conjunto de treino, o que pode levar a uma precisão inferior do classificador quando aplicado ao conjunto de teste. Isso pode explicar também alguns valores inferiores para a taxa de acerto com os subconjuntos selecionados com o CDM com  $n = 80$  quando comparado com o CDM com  $n = 1$ . A proximidade entre as soluções ótimas e subótimas é análoga àquela observada nos gráficos das Figuras 5.4 e 5.5. Ou seja, os subconjuntos selecionados com o CDM com  $n = 80$  são mais próximos dos ótimos do que os selecionados com a outra função critério. Não foi realizada a comparação entre as soluções ótimas e subótimas da seleção com a distância de JM pois o SFFS encontrou todas as soluções ótimas.

A comparação de todos os resultados subótimos são apresentados no gráfico da Figura 5.10. De acordo com o que é mostrado no gráfico, os melhores subconjuntos foram selecionados com o CDM com  $n = 1$  e os piores com a distância de JM. Espera-se que uma função critério dependente leve a melhores resultados. Porém, um estimador mais exato, como o CDM com  $n = 80$ , deveria levar a melhores resultados. A falta de exatidão do *holdout* e uma seleção muito específica para o conjunto de treino podem explicar tal resultado. Ainda assim, as duas funções critério dependentes levaram a taxas de acerto próximas e uma queda quando  $d > 30$ , o que comprova a vantagem da seleção de características. A Tabela 5.2 mostra o número de características de cada método de

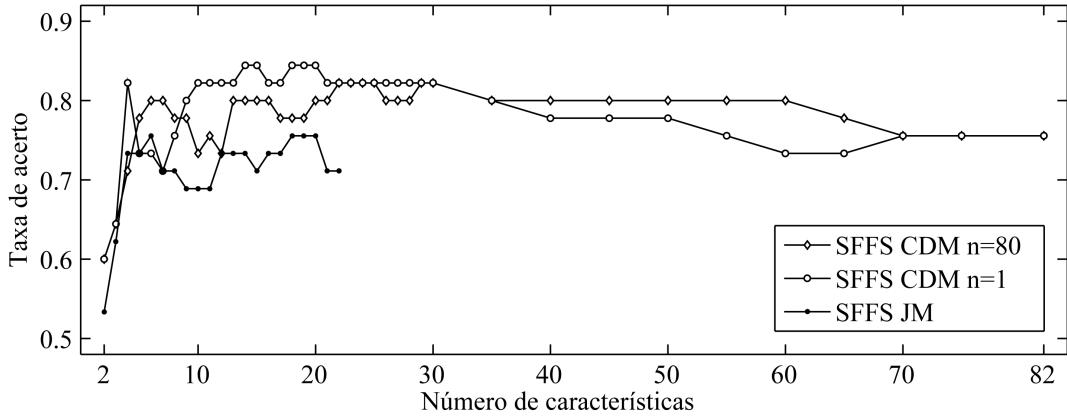


Figura 5.10: Taxa de acerto obtida na validação em relação ao número de características selecionadas com o SFFS. O resultado para  $d = 82$  foi incluído apenas para referência, o subconjunto não foi obtido com uma seleção.

Função critério	Taxa de acerto na validação	$d$	Método de extração		
			Matrizes de co-ocorrência	Filtros de Gabor	Primeira ordem
Distância de JM	75,56%	6	1	4	1
CDM $n = 1$	84,44%	14	4	7	3
CDM $n = 80$	82,22%	22	8	13	1

Tabela 5.2: Número de características de cada método de extração dos subconjuntos que atingiram a maior taxa de acerto na validação.

extração dos subconjuntos que atingiram a maior taxa de acerto na validação. Observa-se que as características de Gabor são predominantes nos três subconjuntos. Em seguida estão as características de matrizes de co-ocorrência. Além disso, a presença dos três métodos nos casos apresentados reforça a idéia de que há vantagens em se utilizar métodos diferentes combinadamente.

## 5.2 Segmentação de Mosaicos

A segmentação de mosaicos é freqüentemente utilizada para testes em análise de imagens (Gerhardinger, 2006; Santos, 2007; Tuceryan e Jain, 1998). Os mosaicos empregados nesse experimento são formados pela combinação de 4 texturas de Brodatz (Brodatz, 1966 apud Tuceryan e Jain, 1998). As imagens originais possuem 256 tons de cinza e  $512 \times 512$  píxeis. Duas regiões com  $256 \times 256$  píxeis sem sobreposição foram extraídas de cada imagem. Na Figura 5.11a, pode-se observar uma das imagens originais com as regiões demarcadas por linhas pretas. As regiões obtidas do canto superior esquerdo foram utilizadas para formar um mosaico para a seleção de características e as regiões do canto superior direito foram utilizadas em dois mosaicos para validação. As 8 regiões podem ser observadas na Figura 5.11b. Os mosaicos gerados são mostrados na Figura 5.12.

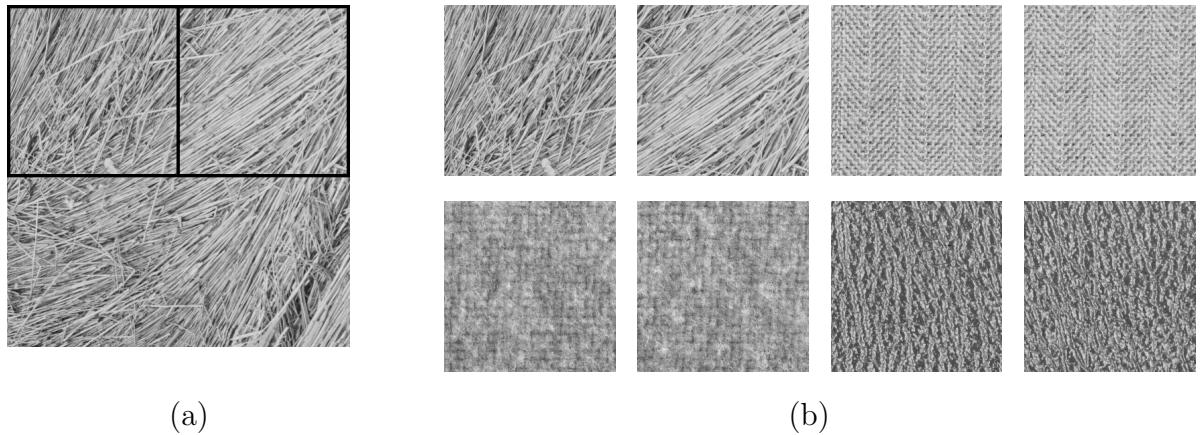


Figura 5.11: Texturas de Brodatz usadas para formar os mosaicos. (a) Uma das imagens originais com as duas regiões  $256 \times 256$  demarcadas por linhas pretas. (b) As 8 regiões extraídas.

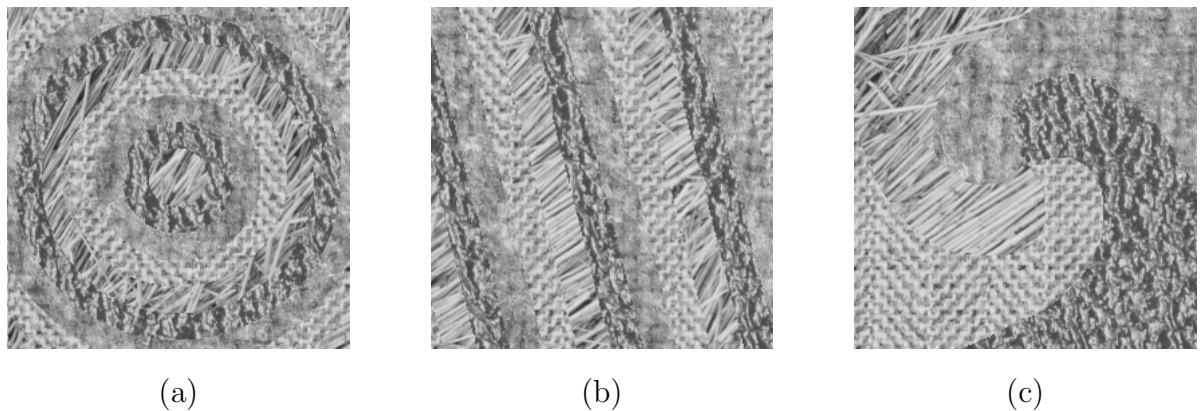
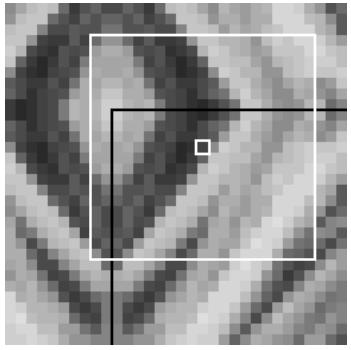


Figura 5.12: Mosaicos utilizados no experimento. (a) Imagem empregada na seleção de características. (b) e (c) Imagens empregadas na validação.

Foram extraídas 108 características de cada píxel dos mosaicos, sendo que 48 foram obtidas com matrizes de co-ocorrência, 48 com filtros de Gabor e 12 com estatísticas de primeira ordem. No caso das características de matrizes de co-ocorrência e de primeira ordem, a extração foi feita de uma região ao redor dos píxeis delimitada por janelas quadradas de lado  $s = 21$  e  $s = 31$ . Nesse tipo de abordagem, parte da janela pode ser posicionada fora da área da imagem. Isso foi tratado pela expansão de  $e = (s - 1)/2$  píxeis de cada lado da imagem. Para preservar a textura na região expandida, parte da própria imagem foi replicada, porém, de maneira invertida. A Figura 5.13 mostra o canto superior esquerdo de uma das imagens após a expansão de  $e = 10$  píxeis. Observa-se também uma janela de lado  $s = 21$  (representada pelo quadrado branco maior) centralizada em relação ao píxel correspondente (indicado pelo quadrado branco menor). A matriz **I** da Figura 5.13 mostra os valores dos píxeis de uma outra imagem e a matriz **I'** mostra o resultado da expansão de  $e = 2$  píxeis dessa imagem. No caso das características de Gabor, as características de cada píxel são os próprios valores obtidos com a convolução e o cálculo da magnitude, como apresentado na Seção 2.4.3.



$$\mathbf{I} = \begin{bmatrix} 5 & 2 & 0 & 2 & \dots \\ 3 & 1 & 3 & 1 & \dots \\ 7 & 1 & 2 & 5 & \dots \\ 4 & 8 & 2 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad \mathbf{I}' = \begin{bmatrix} 1 & 3 & 3 & 1 & 3 & 1 & \dots \\ 2 & 5 & 5 & 2 & 0 & 2 & \dots \\ 2 & 5 & 5 & 2 & 0 & 2 & \dots \\ 1 & 3 & 3 & 1 & 3 & 1 & \dots \\ 1 & 7 & 7 & 1 & 2 & 5 & \dots \\ 8 & 4 & 4 & 8 & 2 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Figura 5.13: Exemplos de expansão de imagens. À esquerda, o canto superior esquerdo de uma das imagens após a expansão de  $e = 10$  pixels. A linha preta mostra o limite da imagem antes da expansão, o quadrado branco maior representa uma janela  $21 \times 21$  e o quadrado branco menor indica o pixel correspondente a essa janela. No restante da figura, outra imagem representada na matriz  $\mathbf{I}$  e a expansão de  $e = 2$  pixels dessa mesma imagem representada na matriz  $\mathbf{I}'$ .

Os parâmetros empregados na extração de cada característica podem ser vistos nas Tabelas Apêndice A.4, Apêndice A.5 e Apêndice A.6. O número de cores das imagens foi reduzido para 32 apenas para geração das matrizes de co-ocorrência. Para os demais métodos, o número de cores da imagem original foi preservado. Foram geradas 4 matrizes para cada tamanho de janela ( $21 \times 21$  e  $31 \times 31$ ) e as 6 funções apresentadas na Seção 2.1.2 foram aplicadas em cada matriz. Os filtros de Gabor foram gerados utilizando-se  $S = 8$ ,  $K = 6$ ,  $U_l = 0,02$  e  $U_h = 0,3$ . As 6 funções de estatística de primeira ordem apresentadas na Seção 2.1.1 foram aplicadas em cada região obtida com as duas janelas ( $21 \times 21$  e  $31 \times 31$ ) para cada pixel. Os tamanhos de janela e os parâmetros utilizados para extração das características com matrizes de co-ocorrência e filtros de Gabor foram determinados empiricamente. Assim, de cada imagem de mosaico, obteve-se uma base de dados com 65.536 exemplos, 108 características e 4 classes.

A seleção de características foi realizada com a base de dados do mosaico formado por círculos, presente na Figura 5.12a. A validação foi realizada com a segmentação não-supervisionado dos outros dois mosaicos, mostrados nas Figuras 5.12b e 5.12c. Para a segmentação, a base de dados foi normalizada, dividindo-se cada valor pelo desvio padrão da característica correspondente. O algoritmo *k-means* foi empregado para separação dos exemplos da base de dados em grupos e, com isso, segmentar a imagem, já que cada exemplo da base corresponde a um pixel. A distância euclidiana foi utilizada no *k-means* e a normalização realizada previamente contribuiu para a melhora da precisão do algoritmo. O mosaico formado por círculos de texturas foi reservado para a seleção de características pois apresenta bordas em todas as direções e regiões relativamente pequenas para as diversas texturas. O uso de uma base de dados adequada na seleção é importante, pois as características são escolhidas de acordo com as propriedades de tal base. O uso de um mosaico com regiões grandes iria, provavelmente, reduzir a escolha de características para

separação de detalhes da imagem. O número de exemplos da base de dados é maior do que o necessário para a seleção de características. Portanto, apenas um conjunto de 5.000 exemplos escolhidos aleatoriamente foi empregado nessa etapa do experimento.

A distância de JM foi utilizada como função critério independente. Não é possível, a rigor, utilizar uma função critério dependente, pois a validação não é supervisionada. Entretanto, a taxa de acerto do CDM com distância euclidiana normalizada também foi empregada, pois o mecanismo desse algoritmo é semelhante ao do *k-means*. A estimativa da taxa de acerto foi feita com o *10-fold cross-validation* sem repetições. Não houve a necessidade de repetições, pois o número de exemplos já é grande o suficiente para uma boa estimativa da precisão do classificador.

Para a distância de JM, a seleção foi realizada com diferentes versões do *branch and bound*, a busca exaustiva e o SFFS. Com exceção do SFFS, todos os outros algoritmos encontram a solução ótima sempre. A finalidade de incluí-los no experimento é apenas para comparar o tempo de processamento. No caso da taxa de acerto do CDM, os algoritmos utilizados foram a busca exaustiva e o SFFS.

As soluções ótimas da distância de JM foram obtidas para subconjuntos de características de tamanhos entre 1 e 6, enquanto que as soluções subótimas foram obtidas para todos os tamanhos entre 2 e 29 e de 5 em 5 entre 30 e 105 ( $30, 35, 40, \dots, 105$ ). A comparação entre os valores ótimos e subótimo (encontrados com o SFFS) podem ser vistos na Figura 5.14. De acordo com o gráfico, o SFFS aparentemente encontrou as soluções ótimas em todos os casos para os tamanhos entre 2 e 6. Porém, apenas para os tamanhos 2 e 4 isso ocorreu. Para os outros tamanhos, soluções muito próximas das ótimas foram encontradas. Por exemplo, o SFFS encontrou  $J(\{40, 41, 103, 107, 108\}) = 1,73886$ , enquanto que a solução ótima é  $J(\{28, 41, 103, 107, 108\}) = 1,73895$ . Nesse caso, pode-se notar também que essas soluções possuem apenas uma característica diferente. Sendo que as duas características correspondem à variância de matrizes de co-ocorrência obtidas com janelas  $31 \times 31$ , uma para o deslocamento  $(1, 0)$  e a outra para o deslocamento  $(2, 0)$ . Portanto, são subconjuntos muito parecidos e que, consequentemente, levam a valores próximos da função critério, o que justifica a resposta do SFFS.

A comparação entre os resultados ótimos e subótimos obtidos com a taxa de acerto do CDM é apresentada na Figura 5.15. As soluções ótimas foram determinadas apenas para subconjunto de tamanho entre 1 e 4 devido ao alto tempo de processamento necessário para subconjuntos maiores. As soluções subótimas foram obtidas com o SFFS para subconjuntos de tamanho entre 2 e 105, seguindo a mesma seqüência de tamanhos empregada com a distância de JM. O SFFS encontrou a solução ótima para o subconjunto de 2 características e chegou muito perto para subconjuntos de 3 e 4 características. Observa-se também que há pouca oscilação na curva resultante. Isso indica que a precisão do CDM foi estimada com boa exatidão, o que contribui para o SFFS encontrar a solução ótima. Em relação a todos os tamanhos de subconjunto, o maior valor da função critério

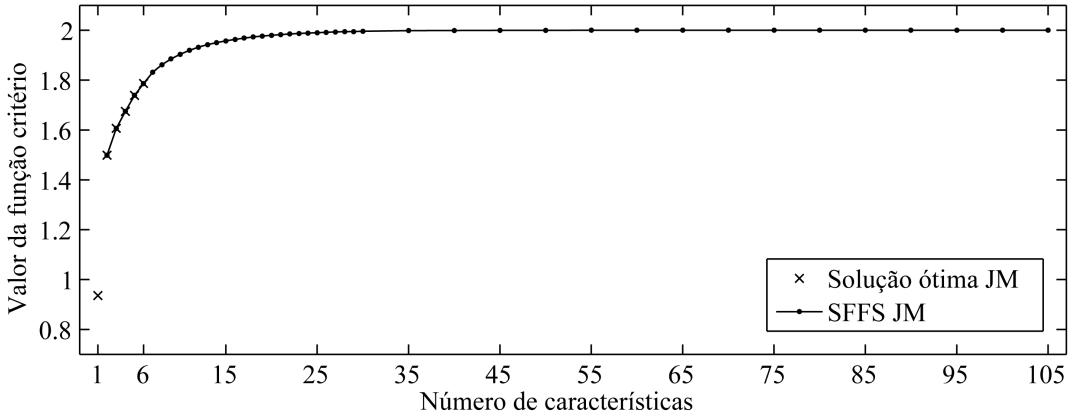


Figura 5.14: Distância de JM em relação ao número de características selecionadas.

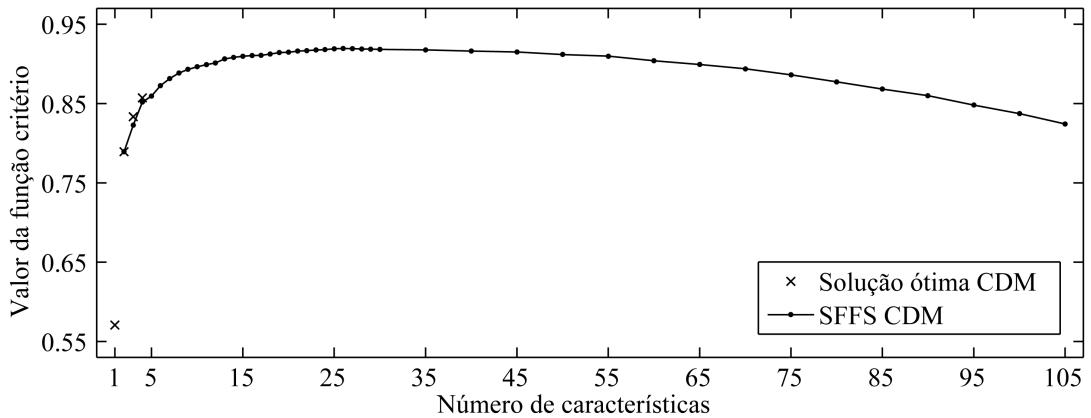


Figura 5.15: Taxa de acerto média do CDM em relação ao número de características selecionadas.

encontrado foi  $J(\cdot) = 0,9194$  quando  $d = 26$ . A diminuição do valor, à medida que o número de características aumenta, mostra um ganho de precisão com a redução da dimensionalidade.

Alguns dos tempos de processamento dos métodos analisados podem ser vistos na Tabela 5.3. A Figura 5.16 mostra um gráfico semilog para os tempos de parte dos métodos. Não foram incluídos todos os métodos no gráfico porque algumas linhas ficariam sobrepostas, prejudicando a visualização. Pode-se notar que o cálculo da distância de JM é muito mais rápido do que a estimativa da precisão do CDM, mesmo com a redução do número de exemplos da base de dados. Em muitas situações, o BB apresentou desempenho inferior à busca exaustiva. O BB rápido foi mais lento para todos os tamanhos de subconjunto, o BB adaptativo foi sutilmente mais rápido do que a busca exaustiva para  $d = 5$  e  $d = 6$ , enquanto que o BB adaptativo com a estratégia floresta foi consideravelmente mais rápido nos mesmos casos. Em geral, a baixa velocidade do BB é causada por valores altos da função critério em nós distantes das folhas. Com isso, há um menor número de podas na árvore de busca e, consequentemente, um maior número de chamadas

	1	2	3	4	5	6	105
Busca exaustiva CDM	0,1954s	7,4s	327,7s	10.271,4s = 2,9h	—	—	—
Busca exaustiva JM	0,0005s	0,019s	1,0s	38,3s	1.166,1s	28.380,0s = 7,9h	—
BB rápido JM	—	4,5s	7,2s	78,2s	1.738,2s	35.430,8s = 9,8h	—
BB adaptativo JM	—	50,2s	51,1s	93,4s	1.187,3s	23.737,9s = 6,6h	—
BB adaptativo floresta JM	—	53,3s	54,3s	89,5s	923,2s	16.212,2s = 4,5h	—
SFFS CDM	—	2,9s	4,6s	4,8s	5,4s	5,7s	484,8s
SFFS JM	—	0,017s	0,022s	0,060s	0,067s	0,077s	241,7s

Tabela 5.3: Tempo de execução da seleção de características. As colunas correspondem ao número de características selecionadas e as linhas, aos métodos de seleção.

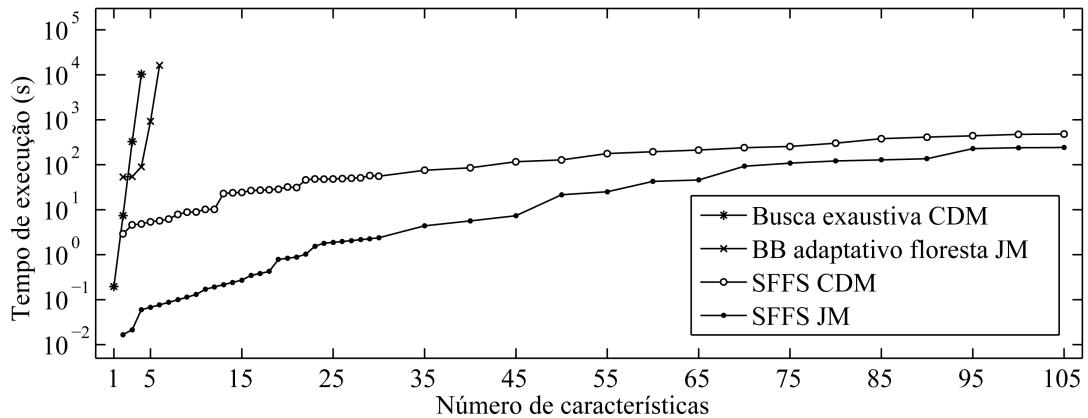


Figura 5.16: Gráfico semilog do tempo de execução da seleção em relação ao número de características selecionadas.

da função critério. O desempenho do BB adaptativo é prejudicado também pelas etapas iniciais do algoritmo. Isso é fácil de perceber observando-se os tempos gastos na seleção de subconjuntos com 2 e 3 características. De fato, a etapa de ordenação das características leva cerca de 50s para qualquer valor de  $d$ . Porém, essa demora é significativa apenas na seleção de subconjuntos pequenos. Espera-se que o BB seja proporcionalmente mais rápido do que a busca exaustiva para  $d > 6$ . Entretanto, o tempo de processamento faz com que o método seja inviável nesses casos.

A validação consistiu apenas na análise da segmentação obtida com o uso de cada subconjunto de características. A imagem empregada na seleção de características também foi segmentada. Porém, nesse caso, o experimento não pode ser considerado como uma reprodução de um uso real da seleção de características, já que a mesma base de dados foi utilizada em todas as etapas. Os resultados de algumas segmentações podem ser vistos na Tabela 5.4. A segmentação obtida com 4 subconjuntos e o conjunto completo

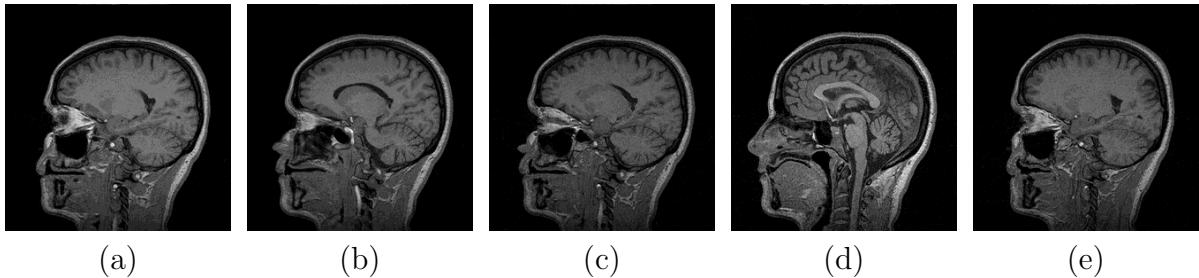


Figura 5.17: Algumas das imagens da base. (a) Imagem empregada na seleção de características e treino do CDM. (b)–(e) Imagens empregadas na validação.

são apresentados. Dos subconjuntos, 2 foram selecionados com a distância de JM e os outros 2, com a taxa de acerto do CDM. A precisão da segmentação, mostrada abaixo das imagens, foi determinada pela comparação com os gabaritos, também ilustrados na Tabela 5.4. As segmentações mostradas não são necessariamente as que atingiram a maior taxa de acerto, mas as que atingiram uma taxa de acerto próxima da maior com um número reduzido de características. Portanto, pode-se concluir que foi possível um aumento da precisão da segmentação com a redução da dimensionalidade. Além disso, a utilização da taxa de acerto do CDM como função critério permitiu uma melhor seleção de características, mesmo não sendo o mesmo algoritmo empregado na segmentação. Assim como no experimento da Seção 5.1, quase todos os subconjuntos selecionados contém características dos três métodos de extração empregados.

### 5.3 Segmentação de Imagens Médicas

Nesse experimento, foi utilizado um conjunto de 38 imagens MRI do cérebro. As imagens possuem  $256 \times 256$  píxeis e 256 tons de cinza. O problema abordado foi a segmentação supervisionada das imagens, com o objetivo de identificar três regiões: (1) cérebro, (2) restante da cabeça e (3) fundo. Para treino e posterior validação, as imagens foram inicialmente segmentadas manualmente. A imagem mostrada na Figura 5.17a foi empregada na validação e seleção de características. Essa imagem foi escolhida arbitrariamente. Algumas das outras imagens da base são mostradas nas Figuras 5.17b–5.17e.

Foram extraídas 130 características de cada píxel das imagens, sendo que 72 foram obtidas com matrizes de co-ocorrências, 40 com filtros de Gabor e 18 com estatísticas de primeira ordem. No caso das características de matrizes de co-ocorrência e de primeira ordem, foram utilizadas janelas de lado  $s = 7$ ,  $s = 11$  e  $s = 15$ . O processo de expansão das imagens para utilização de janelas foi o mesmo empregado na segmentação de mosaicos, como explicado na Seção 5.2. As características de Gabor também foram obtidas pelo mesmo processo do experimento anterior, que consiste apenas na convolução dos filtros e cálculo da magnitude.

Gabarito	Distância de JM	Taxa de acerto do CDM	Todas as caract.
—	$\{40, 41, 103, 107, 108\}$	$\{2, 20, 25, 37, 39, 40, 41, 42, 82, 84, 86, 103, 107, 108\}$	$Y = \{1, 2, \dots, 108\}$

Os parâmetros empregados na extração de cada característica podem ser vistos nas Tabelas Apêndice A.7, Apêndice A.8 e Apêndice A.9. O número de cores das imagens foi reduzido para 32 apenas para extração das características de matrizes de co-ocorrência. Foram geradas 4 matrizes para cada tamanho de janela e as 6 funções apresentadas na Seção 2.1.2 foram aplicadas em cada matriz. Os filtros de Gabor foram gerados utilizando-se  $S = 20$ ,  $K = 2$ ,  $U_l = 0,05$  e  $U_h = 0,35$ . As características de primeira ordem foram extraídas com a aplicação das 6 funções apresentadas na Seção 2.1.1 nas regiões obtidas com cada tamanho de janela em cada píxel. Os tamanhos de janela e os parâmetros utilizados para extração das características com matrizes de co-ocorrência e filtros de Gabor foram determinados empiricamente. Assim, de cada imagem médica, obteve-se uma base de dados com 65.536 exemplos, 130 características e 3 classes.

O uso de apenas uma imagem para treino é, a rigor, muito pouco. Porém, verificou-se que, mesmo com um conjunto reduzido de treino, é possível atingir uma precisão razoável com uma segmentação supervisionada em muitas das outras imagens da base. O pequeno número de classes e a existência de uma classe bem contrastante (o fundo) facilitam o reconhecimento de padrões. O CDM com distância euclidiana normalizada foi empregado na segmentação. Para a seleção de características, o conjunto de treino foi reduzido para 5.000 exemplos escolhidos aleatoriamente.

A distância de Bhattacharyya não pôde ser empregada na avaliação de subconjuntos nesse experimento. A causa desse impedimento foi a existência de características aproximadamente constantes na base de dados (algumas das características dos píxeis do fundo da imagem). Portanto, nenhuma função critério independente foi empregada. Como função critério dependente, foi utilizada a taxa de acerto do CDM estimada com o *10-fold cross-validation* sem repetições. Não houve a necessidade de repetições devido ao grande número de exemplos no conjunto de treino.

A busca exaustiva foi utilizada para seleção de uma característica e o SFFS para seleção de outros tamanhos de subconjunto. O objetivo desse experimento não foi comparar diferentes métodos de seleção. Optou-se por fazer a seleção com o conjunto completo de características e com os conjuntos formados pelas características de cada método de extração isoladamente. Portanto, foram comparados os resultados (1) da seleção de todas as características obtidas, (2) da seleção das características de matrizes de co-ocorrência, (3) da seleção das características de Gabor e (4) da seleção das características de primeira ordem.

Os valores da função critério obtidos para cada subconjunto selecionado podem ser vistos no gráfico da Figura 5.18. Do conjunto completo de características, foram selecionados subconjuntos de tamanho  $d = 1, 2, \dots, 40, 50, \dots, 130$ . Do conjunto das características de matrizes de co-ocorrência, foram selecionados subconjuntos de tamanho  $d = 1, 2, \dots, 40, 50, 60, 70, 72$ . Do conjunto das características de Gabor, foram selecionados subconjuntos de tamanho  $d = 1, 2, \dots, 40$ . Do conjunto das características de primeira

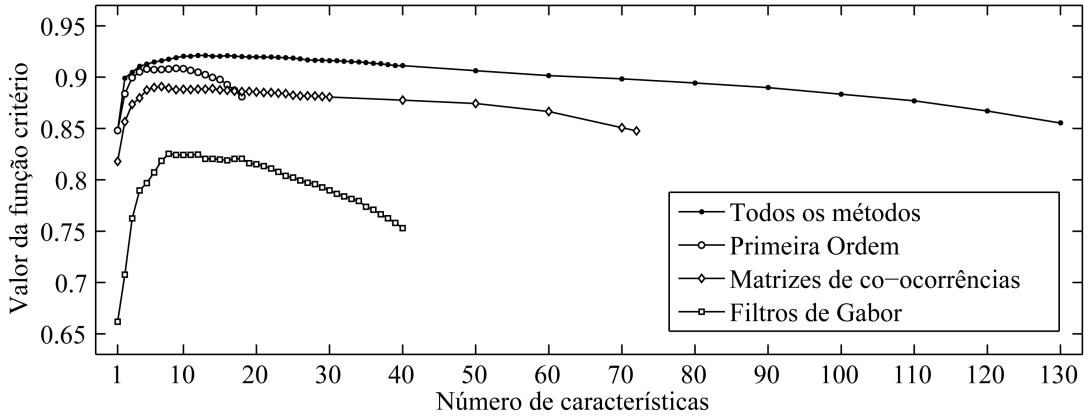


Figura 5.18: Taxa de acerto do CDM em relação ao número de características selecionadas. As características foram extraídas da imagem médica de treino.

ordem, foram selecionados subconjuntos de tamanho  $d = 1, 2, \dots, 18$ . Os subconjuntos com apenas características de Gabor resultaram em uma taxa de acerto significantemente inferior aos demais. As taxas de acerto mais altas foram obtidas com subconjuntos do conjunto completo de características. Os subconjuntos de todas as características devem resultar em valores mais altos pois todas as características estão disponíveis para seleção. De fato, as características predominantes nos melhores subconjuntos são de primeira ordem, que também atingiram valores altos isoladamente. Portanto, pode-se concluir que esse tipo de imagem é melhor segmentado com características de primeira ordem, caso seja utilizado apenas um dos métodos de extração de características.

Para validar o método de seleção empregado, as imagens restantes da base foram segmentadas utilizando-se os subconjuntos selecionados. A taxa de acerto foi calculada pela comparação com a segmentação manual realizada previamente. Apesar de a segmentação manual ser subjetiva, a diferenciação entre as partes das imagens é suficientemente clara para um observador humano e foi a única maneira encontrada para determinar a taxa de acerto do classificador. Os resultados da validação podem ser vistos no gráfico da Figura 5.19. Observa-se que o comportamento desse gráfico aproxima-se do comportamento do gráfico da Figura 5.18. Entretanto, alguns subconjuntos com apenas características de primeira ordem atingiram taxas de acerto maior do que os demais. A principal causa disso é diferença entre a imagem de treino e as imagens de teste. Isto é, as características selecionadas são específicas para a imagem de treino e não são necessariamente as melhores para outras imagens. Além disso, o bom resultado obtido com as características de primeira ordem confirma que esse método de extração é adequado ao programa abordado. Mesmo assim, as taxas de acerto dos subconjuntos selecionados a partir do conjunto completo são muito próximas dos maiores valores obtidos. Logo, não é possível afirmar que as características de primeira ordem devem ser usadas isoladamente nesse caso.

Os subconjuntos que atingiram as maiores taxas de acerto para cada conjunto

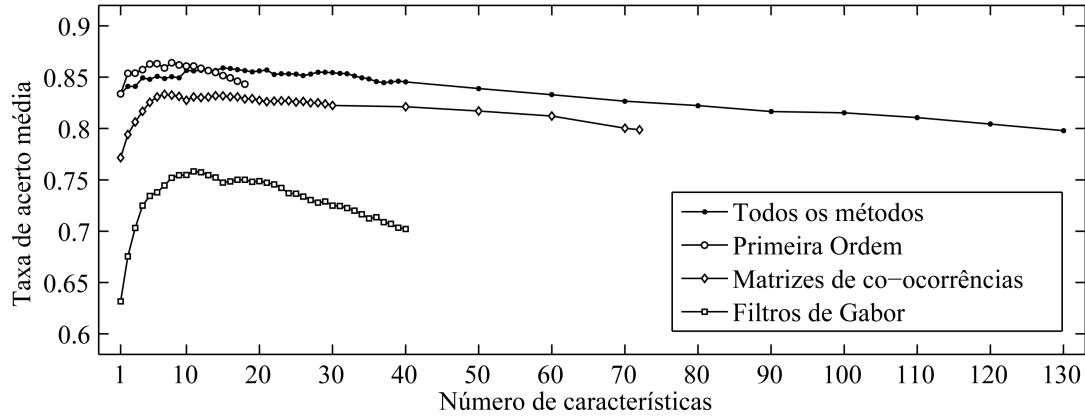


Figura 5.19: Taxa de acerto obtida na validação em relação ao número de características selecionadas.

inicial de características foram utilizados para a construção da Tabela 5.5. A segmentação obtida com o conjunto completo de características também é mostrada, para ilustrar o ganho em precisão com a redução da dimensionalidade. Os resultados obtidos com as outras imagens são semelhantes aos apresentados na tabela. Levando-se em conta apenas esses subconjuntos utilizados na tabela, 15 imagens foram mais bem segmentadas com o subconjunto de características dos três métodos, 14 foram mais bem segmentadas apenas com características de primeira ordem e 8 imagens foram mais bem segmentadas apenas com características de matrizes de co-ocorrência. Pode-se observar também que, do subconjunto com características dos três métodos, 3 são características de matrizes de co-ocorrência, 5 são características de Gabor e 7 são características de primeira ordem.

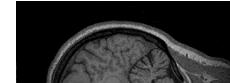
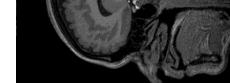
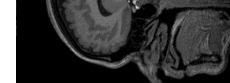
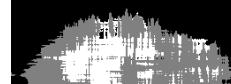
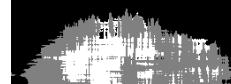
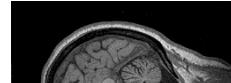
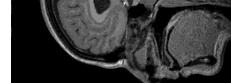
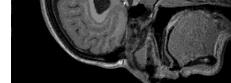
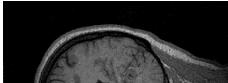
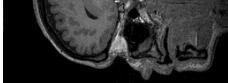
Imagen original	Todos os métodos de extração de características	Matrizes de co-ocorrência	Filtros de Gabor	Primeira ordem
$\{22, 40, 53, 73, 78, 80,$ $81, 83, 113, 114, 117,$ $118, 119, 125, 129\}$	$Y = \{1, 2, \dots, 130\}$	$\{4, 5, 10, 16,$ $22, 56, 63\}$	$\{73, 75, 77, 78,$ $80, 81, 86, 103,$ $106, 107, 108\}$	$\{113, 117, 118, 119,$ $123, 125, 126, 129\}$
				
				
				
				
				
				
				
				
				
				
				

Tabela 5.5: Alguns resultados da segmentação das imagens médicas.



## Conclusões e Trabalhos Futuros

---

---

Neste trabalho, diversos métodos de seleção de características foram estudados. O *branch and bound* e a busca exaustiva foram escolhidos como algoritmos ótimos de seleção para serem empregados nos experimentos. O *branch and bound* foi escolhido por ser tradicionalmente utilizado quando se deseja a solução ótima. Porém, é restrito a funções critério monotônicas, por isso a busca exaustiva também foi empregada. Como algoritmo subótimo, o SFFS foi escolhido por ser amplamente utilizado e apresentar bons resultados em comparações já realizadas. A distância de JM foi empregada como função critério independente também por ser amplamente utilizada em trabalho de reconhecimento de padrões. Como função critério dependente, a taxa de acerto do CDM foi empregada pois o algoritmo é rápido, simples e bem adaptável ao processo de seleção de características.

Diversas versões do *branch and bound* foram estudadas e implementadas. Algumas deficiências foram identificadas e uma nova estratégia, nomeada floresta, foi proposta. Em experimentos realizados, verificou-se que a eficiência do *branch and bound* melhora significativamente com o uso da estratégia floresta.

Os métodos de seleção foram avaliados em problemas que envolviam características de texturas. Diversos métodos de extração de características de texturas de imagens foram estudados. Para os experimentos, foram escolhidos métodos amplamente utilizados e com abordagens diferentes: estatísticas de primeira ordem, matrizes de co-ocorrência e filtros de Gabor. Diferentes tipos de problemas e imagens foram utilizados: (1) classificação de regiões de uma foto aérea de plantação de eucalipto; (2) segmentação não-supervisionada de mosaicos de texturas de Brodatz e (3) segmentação supervisionada de imagens médicas (MRI do cérebro).

A distância de JM ficou limitada à seleção de subconjuntos pequenos de características ( $d \leq 22$ ) no experimento (1) e não pôde ser empregada no experimento (3). O

número reduzido de exemplos no experimento (1) e características de valores aproximadamente constantes no experimento (3) foram as causas dessas restrições. Portanto, o *branch and bound* pôde ser utilizado apenas no experimento (2).

A existência de limitações para o uso da distância de JM é freqüente em seleção de características de textura. Os subconjuntos selecionados com essa função critério levaram a taxas de acerto inferiores àquelas obtidas com subconjuntos selecionados com a função critério dependente. Mesmo quando um algoritmo diferente do CDM foi empregado na segmentação, a taxa de acerto obtida foi superior. Portanto, uma função critério dependente é desejável em problemas desse tipo. Em situações em que o algoritmo de classificação é muito lento para ser empregado na função critério, o CDM pode ser considerado.

A menos que se deseje selecionar poucas características ( $d < 5$ ) ou que o tempo de processamento não seja relevante, os métodos ótimos de seleção não são viáveis. O *branch and bound* é mais eficiente do que a busca exaustiva em muitos casos, tendo a estratégia floresta contribuído para uma maior eficiência. Mesmo assim, o tempo necessário para processamento é muito grande. Os melhores subconjuntos freqüentemente possuem mais do que 5 características, o que torna necessário o uso de um método subótimo para seleção.

O SFFS selecionou subconjuntos iguais ou muito próximos aos ótimos na maioria das situações analisadas. O tempo de execução desse algoritmo foi satisfatório em todos os casos. Assim, pode-se concluir que, dos métodos avaliados, o melhor foi o SFFS com função critério dependente, em relação à qualidade dos subconjuntos selecionados, ao tempo de execução e à variedade de problemas em que pode ser empregado.

Em todos os experimentos, a redução da dimensionalidade melhorou a precisão no reconhecimento de padrões. A utilização de seleção de características para diminuir o número de características de uma base de dados ainda contribui para diminuir o custo computacional da extração de características. No caso de características de texturas, a seleção ainda pode ser empregada na avaliação de quais são os melhores métodos e parâmetros para a extração de características de determinado tipo de imagem. Porém, a utilização conjunta de diferentes métodos é mais apropriada do que a utilização de um método isoladamente, como foi confirmado nos experimentos.

Como trabalhos futuros, muitos outros métodos de seleção ainda podem ser avaliados. Existem outras abordagens e algoritmos de busca, como redes neurais e algoritmos genéticos, e outras funções critério, como a utilização de outros classificadores em funções dependentes. O *branch and bound* ainda pode ser melhorado. A estratégia floresta pode ser modificada, com a inclusão da reordenação das características em algumas subárvores e uma outra maneira para a geração das árvores. Porém, dificilmente o *branch and bound* ficará rápido o suficiente para ser viável na seleção de muitas características ( $D > 30$ ). Outras abordagens podem ser empregadas para encontrar a solução ótima, como o cálculo recursivo da função critério na busca exaustiva e a utilização de computação paralela.

# Referências

---

---

- ALBREGTSEN, F.; NIELSEN, B.; DANIELSEN, H. Adaptive gray level run length features from class distance matrices. *International Conference on Pattern Recognition*, v. 3, p. 738–741, 2000.
- BANKS, S. *Signal processing, image processin and pattern recognition*. Prentice Hall, 1990.
- BARALDI, A.; PARMIGGIANI, F. An investigation of the textural characteristics associated withgray level cooccurrence matrix statistical parameters. *IEEE Transactions on Geoscience and Remote Sensing*, v. 33, n. 2, p. 293–304, 1995.
- BELLMAN, R. *Adaptive control processes: A guided tour*. Princeton University Press, 1961.
- BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B. *Redes neurais artificiais: Teoria e aplicações*. LTC, 2000.
- BRIGHAM, O. *The fast Fourier transform*. Prentice-Hall, 1974.
- CASTELLANO, G.; FANELLI, A. M. Variable selection using neural-network models. *Neurocomputing*, v. 31, n. 1-4, p. 1–13, 2000.
- CHEN, X. An improved branch and bound algorithm for feature selection. *Pattern Recognition Letters*, v. 24, n. 12, p. 1925–1933, 2003.
- CLARK, M.; BOVIK, A. C.; GEISLER, W. S. Texture segmentation using Gabor modulation/demodulation. *Pattern Recognition Letters*, v. 6, n. 4, p. 261–267, 1987.
- COMER, M. L.; DELP, E. J. The EM/MPM algorithm for segmentation of textured images: Analysis and further experimental results. *IEEE Trans. Image Processing*, v. 9, n. 10, p. 1731–1744, 2000.
- DAUGMAN, J. Gabor wavelets and statistical pattern recognition. In: ARBIB, M. A., ed. *The Handbook of Brain Theory and Neural Networks*, 2º ed, MIT Press, p. 457–463, 2003.
- DAUGMAN, J. G. Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 36, n. 7, p. 1169–1179, 1988.

- DAUGMAN, J. G.; DOWNING, C. Gabor wavelets for statistical pattern recognition. In: ARBIB, M. A., ed. *The Handbook of Brain Theory and Neural Networks*, Cambridge, Massachusetts: MIT Press, p. 414–419, 1995.
- FERRARI, R.; RANGAYYAN, R.; DESAUTELS, J.; BORGES, R.; FRÈRE, A. Automatic identification of the pectoral muscle in mammograms. *IEEE Transactions on Medical Imaging*, v. 23, n. 2, p. 232–245, 2004.
- FERRI, F.; PUDIL, P.; HATEF, M.; KITTLER, J. Comparative study of techniques for large-scale feature selection. In: GELSEMA, E. S.; KANAL, L. N., eds. *Pattern Recognition in Practice IV*, Amsterdam: Elsevier Science Inc., 1994, p. 403–413.
- FUKUNAGA, K. *Introduction to pattern recognition*. 2º ed. Academic Press, 1990.
- GABOR, D. Theory of communication. *Journal of Electrical Engineers*, v. 93, p. 429–457, 1946.
- GALLOWAY, M. M. Texture analysis using gray level run lengths. *Computer Graphics Image Processing*, v. 4, p. 172–179, 1975.
- GARSON, G. D. Interpreting neural net connection weights. *AI Expert*, v. 6, n. 4, p. 46–51, 1991.
- GERHARDINGER, L. C. *Segmentação de imagens e validação de classes por abordagem estocástica*. Dissertação de Mestrado, Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 2006.
- GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*. Addison-Wesley Publishing Company, 1992.
- HARALICK, R. M.; SHANMUGAN, K. S.; DUNSTEIN, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 3, n. 6, p. 610–621, hARALICK73, 1973.
- HAYKIN, S. *Neural networks: A comprehensive foundation*. 2º ed. Prentice Hall, 1999.
- HE, D. C.; WANG, L. Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, v. 28, n. 4, p. 509–512, 1990.
- HE, D. C.; WANG, L. Texture features based on texture spectrum. *Pattern Recognition*, v. 24, n. 5, p. 391–399, 1991.
- JAIN, A.; ZONGKER, D. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 2, p. 153–158, 1997.
- JULESZ, B. Visual pattern discrimination. *IEEE Transactions on Information Theory*, v. 8, n. 2, p. 84–92, 1962.
- KLIMANEE, C.; NGUYEN, D. On the design of 2-D Gabor filtering of fingerprint images. In: *Proceedings of 2004 IEEE Consumer Communications and Networking Conference (CCNC2004)*, Las Vegas, Nevada, USA, 2004, p. 430–435.

- KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, v. 97, p. 273–323, 1997.
- KUDO, M.; SKLANSKY, J. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, v. 33, n. 1, p. 25–41, 2000.
- LACERDA, E. G. M. d.; DE CARVALHO, A. C. P. L. F.; LUDELMIR, T. B. Um tutorial sobre algoritmos genéticos. *Revista de Informática Teórica e Aplicada*, v. 9, n. 3, p. 109–139, 2002.
- LEE, C.-J.; WANG, S.-D. A Gabor filter-based approach to fingerprint recognition. In: *IEEE Workshop on Signal Processing Systems, 1999. SiPS 99.*, 1999, p. 371–378.
- LEFEBVRE, L.; POULIN, P. Analysis and synthesis of structural textures. In: *Graphics Interface*, 2000, p. 77–86.
- LIU, H.; YU, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 4, p. 491–502, 2005.
- MANDELBROT, B. B. *The fractal geometry of nature*. W. H. Freeman and Company, 1983.
- MANJUNATH, B. S.; MA, W.-Y. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 18, n. 8, p. 837–842, 1996.
- MATERKA, A.; STRZELECKI, M. *Texture analysis methods - a review*. Relatório Técnico, Technical University of Lodz, Institute of Electronics, 1998.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the idea immanent in nervous activity. *Bulletin of mathematical biophysics*, v. 5, p. 115–133, 1943.
- NAKARIYAKUL, S.; CASASENT, D. P. Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognition Letters*, v. 28, n. 12, p. 1415–1427, 2007.
- NARENDRA, P. M.; FUKUNAGA, K. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, v. 26, n. 9, p. 917–922, 1977.
- NATH, R.; RAJAGOPALAN, B.; RYKER, R. Determining the saliency of input variables in neural network classifiers. *Computers & Operations Research*, v. 24, n. 8, p. 767–773, 1997.
- OJALA, T.; PIETIKÄINEN, M.; HARWOOD, D. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, v. 29, n. 1, p. 51–59, 1996.
- PLOTNICK, R. E.; GARDNER, R. H.; HARGROVE, W. W.; PRESTEGAARD, K.; PERLMUTTER, M. Lacunarity analysis: A general technique for the analysis of spatial patterns. *Physical Review E*, v. 53, n. 5, p. 5461–5468, 1996.
- PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. *Numerical recipes in c: The art of scientific computing*. 2º ed. Cambridge University Press, 1992.

- PUDIL, P.; Novovičová, J.; KITTLER, J. Floating search methods in feature selection. *Pattern Recognition Letters*, v. 15, n. 10, p. 1119–1125, 1994.
- RESENDE, S. O., ed. *Sistemas inteligentes: Fundamentos e aplicações*. Manole, 2003.
- RICHARDS, J. A. *Remote sensing digital image analysis*. 2nd ed. Springer-Verlag, 1993.
- ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, p. 386–408, 1958.
- RUMELHART, D.; McCLELLAND, J. *Parallel distributed processing*, v. 1. Cambridge, MA: MIT Press, 1986.
- SANTOS, D. P. *Seleção de características: Abordagem via redes neurais aplicada à segmentação de imagens*. Dissertação de Mestrado, Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, 2007.
- SARKAR, N.; CHAUDHURI, B. B. An efficient approach to estimate fractal dimension of textural images. *Pattern Recognition*, v. 25, n. 9, p. 1035–1041, 1992.
- SIEDLECKI, W.; SKLANSKY, J. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, v. 10, n. 5, p. 335–347, 1989.
- SILVA, S. *Realimentação de relevantes via algoritmos genéticos aplicada à recuperação de imagens*. Dissertação de Mestrado, Universidade Federal de Uberlândia, 2006.
- SKLANSKY, J. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 8, p. 237–247, 1978.
- SOMOL, P.; PUDIL, P.; FERRI, F. J.; KITTLER, J. Fast branch & bound algorithm in feature selection. In: SANCHEZ B., PINEDA M. J., W. J., ed. *Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000)*, Orlando, Florida, USA: International Institute of Informatics and Systemics (IIIS), 2000, p. 646–651.
- SOMOL, P.; PUDIL, P.; GRIM, J. Branch & bound algorithm with partial prediction for use with recursive and non-recursive criterion forms. In: SINGH, S.; MURSHED, N. A.; KROPATSCH, W. G., eds. *Proceedings of Second International Conference on Advances in Pattern Recognition (ICAPR 2001)*, Rio de Janeiro, Brasil: Springer, 2001, p. 230–239 (*Lecture Notes in Computer Science*, v.2013).
- SOMOL, P.; PUDIL, P.; KITTLER, J. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 7, p. 900–912, 2004.
- TUCERYAN, M.; JAIN, A. K. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 12, n. 2, p. 211–216, 1990.
- TUCERYAN, M.; JAIN, A. K. Texture analysis. In: *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, 1998, p. 235–276.

WALKER, J. S. *A primer on wavelets and their scientific applications.* CRC Press, 1999.

XU, Y.; ZHANG, X. Gabor filterbank and its application in the fingerprint texture analysis. In: *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, 2005, p. 829–831.

YU, B.; YUAN, B. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, v. 26, n. 6, p. 883–889, 1993.

ZHANG, D.; LU, G. Review of shape representation and description techniques. *Pattern Recognition*, v. 37, n. 1, p. 1–19, 2004.

ZHOU, F.; FENG, J. F.; SHI, Q. Y. Texture feature based on local Fourier transform. In: *ICIP (2)*, 2001, p. 610–613.



# Características Utilizadas

---



---

<b>Caract.</b>	$(d_x, d_y)$	<b>Função</b>
1	(1, 0)	energia
2	(1, 0)	contraste
3	(1, 0)	correlação
4	(1, 0)	variância
5	(1, 0)	mom. dif. inv.
6	(1, 0)	entropia
7	(0, 1)	energia
8	(0, 1)	contraste
9	(0, 1)	correlação
10	(0, 1)	variância
11	(0, 1)	mom. dif. inv.
12	(0, 1)	entropia
13	(2, 0)	energia
14	(2, 0)	contraste
15	(2, 0)	correlação
16	(2, 0)	variância
17	(2, 0)	mom. dif. inv.
18	(2, 0)	entropia

<b>Caract.</b>	$(d_x, d_y)$	<b>Função</b>
19	(0, 2)	energia
20	(0, 2)	contraste
21	(0, 2)	correlação
22	(0, 2)	variância
23	(0, 2)	mom. dif. inv.
24	(0, 2)	entropia
25	(3, 0)	energia
26	(3, 0)	contraste
27	(3, 0)	correlação
28	(3, 0)	variância
29	(3, 0)	mom. dif. inv.
30	(3, 0)	entropia
31	(0, 3)	energia
32	(0, 3)	contraste
33	(0, 3)	correlação
34	(0, 3)	variância
35	(0, 3)	mom. dif. inv.
36	(0, 3)	entropia

Tabela Apêndice A.1: Características extraídas da foto aérea com matrizes de co-ocorrência.

<b>Caract.</b>	$\theta$	<i>W</i>
37	0°	0.020
38	45°	0.020
39	0°	0.023
40	45°	0.023
41	0°	0.027
42	45°	0.027
43	0°	0.031
44	45°	0.031
45	0°	0.035
46	45°	0.035
47	0°	0.041
48	45°	0.041
49	0°	0.047
50	45°	0.047
51	0°	0.054
52	45°	0.054
53	0°	0.063
54	45°	0.063
55	0°	0.072
56	45°	0.072

Tabela Apêndice A.2: Características extraídas da foto aérea com filtros de Gabor.

<b>Caract.</b>	$\theta$	<i>W</i>
57	0°	0.083
58	45°	0.083
59	0°	0.096
60	45°	0.096
61	0°	0.111
62	45°	0.111
63	0°	0.128
64	45°	0.128
65	0°	0.147
66	45°	0.147
67	0°	0.170
68	45°	0.170
69	0°	0.196
70	45°	0.196
71	0°	0.226
72	45°	0.226
73	0°	0.260
74	45°	0.260
75	0°	0.300
76	45°	0.300

<b>Caract.</b>	<b>Função</b>
77	média
78	variância
79	energia
80	entropia
81	obliquidade
82	curtose

Tabela Apêndice A.3: Características extraídas da foto aérea com estatísticas de primeira ordem.

<b>Caract.</b>	<i>h</i>	$(d_x, d_y)$	<b>Função</b>
1	21	(1, 0)	energia
2	21	(1, 0)	contraste
3	21	(1, 0)	correlação
4	21	(1, 0)	variância
5	21	(1, 0)	mom. dif. inv.
6	21	(1, 0)	entropia
7	21	(0, 1)	energia
8	21	(0, 1)	contraste
9	21	(0, 1)	correlação
10	21	(0, 1)	variância
11	21	(0, 1)	mom. dif. inv.
12	21	(0, 1)	entropia
13	21	(2, 0)	energia
14	21	(2, 0)	contraste
15	21	(2, 0)	correlação
16	21	(2, 0)	variância
17	21	(2, 0)	mom. dif. inv.
18	21	(2, 0)	entropia
19	21	(0, 2)	energia
20	21	(0, 2)	contraste
21	21	(0, 2)	correlação
22	21	(0, 2)	variância
23	21	(0, 2)	mom. dif. inv.
24	21	(0, 2)	entropia

<b>Caract.</b>	<i>h</i>	$(d_x, d_y)$	<b>Função</b>
25	31	(1, 0)	energia
26	31	(1, 0)	contraste
27	31	(1, 0)	correlação
28	31	(1, 0)	variância
29	31	(1, 0)	mom. dif. inv.
30	31	(1, 0)	entropia
31	31	(0, 1)	energia
32	31	(0, 1)	contraste
33	31	(0, 1)	correlação
34	31	(0, 1)	variância
35	31	(0, 1)	mom. dif. inv.
36	31	(0, 1)	entropia
37	31	(2, 0)	energia
38	31	(2, 0)	contraste
39	31	(2, 0)	correlação
40	31	(2, 0)	variância
41	31	(2, 0)	mom. dif. inv.
42	31	(2, 0)	entropia
43	31	(0, 2)	energia
44	31	(0, 2)	contraste
45	31	(0, 2)	correlação
46	31	(0, 2)	variância
47	31	(0, 2)	mom. dif. inv.
48	31	(0, 2)	entropia

Tabela Apêndice A.4: Características extraídas dos mosaicos com matrizes de co-ocorrência.

<b>Caract.</b>	$\theta$	$f$
49	0°	0.020
50	30°	0.020
51	60°	0.020
52	90°	0.020
53	120°	0.020
54	150°	0.020
55	0°	0.029
56	30°	0.029
57	60°	0.029
58	90°	0.029
59	120°	0.029
60	150°	0.029
61	0°	0.043
62	30°	0.043
63	60°	0.043
64	90°	0.043
65	120°	0.043
66	150°	0.043
67	0°	0.064
68	30°	0.064
69	60°	0.064
70	90°	0.064
71	120°	0.064
72	150°	0.064

Tabela Apêndice A.5: Características extraídas dos mosaicos com filtros de Gabor.

<b>Caract.</b>	$\theta$	$f$
73	0°	0.094
74	30°	0.094
75	60°	0.094
76	90°	0.094
77	120°	0.094
78	150°	0.094
79	0°	0.138
80	30°	0.138
81	60°	0.138
82	90°	0.138
83	120°	0.138
84	150°	0.138
85	0°	0.204
86	30°	0.204
87	60°	0.204
88	90°	0.204
89	120°	0.204
90	150°	0.204
91	0°	0.300
92	30°	0.300
93	60°	0.300
94	90°	0.300
95	120°	0.300
96	150°	0.300

<b>Caract.</b>	$h$	<b>Função</b>
97	21	média
98	21	variância
99	21	energia
100	21	entropia
101	21	obliquidade
102	21	curtose
103	31	média
104	31	variância
105	31	energia
106	31	entropia
107	31	obliquidade
108	31	curtose

Tabela Apêndice A.6: Características extraídas dos mosaicos com estatísticas de primeira ordem.

<b>Caract.</b>	<i>h</i>	( <i>d<sub>x</sub>, d<sub>y</sub></i> )	<b>Função</b>
1	7	(1, 0)	energia
2	7	(1, 0)	contraste
3	7	(1, 0)	correlação
4	7	(1, 0)	variância
5	7	(1, 0)	mom. dif. inv.
6	7	(1, 0)	entropia
7	7	(0, 1)	energia
8	7	(0, 1)	contraste
9	7	(0, 1)	correlação
10	7	(0, 1)	variância
11	7	(0, 1)	mom. dif. inv.
12	7	(0, 1)	entropia
13	7	(2, 0)	energia
14	7	(2, 0)	contraste
15	7	(2, 0)	correlação
16	7	(2, 0)	variância
17	7	(2, 0)	mom. dif. inv.
18	7	(2, 0)	entropia
19	7	(0, 2)	energia
20	7	(0, 2)	contraste
21	7	(0, 2)	correlação
22	7	(0, 2)	variância
23	7	(0, 2)	mom. dif. inv.
24	7	(0, 2)	entropia
25	11	(1, 0)	energia
26	11	(1, 0)	contraste
27	11	(1, 0)	correlação
28	11	(1, 0)	variância
29	11	(1, 0)	mom. dif. inv.
30	11	(1, 0)	entropia
31	11	(0, 1)	energia
32	11	(0, 1)	contraste
33	11	(0, 1)	correlação
34	11	(0, 1)	variância
35	11	(0, 1)	mom. dif. inv.
36	11	(0, 1)	entropia

<b>Caract.</b>	<i>h</i>	( <i>d<sub>x</sub>, d<sub>y</sub></i> )	<b>Função</b>
37	11	(2, 0)	energia
38	11	(2, 0)	contraste
39	11	(2, 0)	correlação
40	11	(2, 0)	variância
41	11	(2, 0)	mom. dif. inv.
42	11	(2, 0)	entropia
43	11	(0, 2)	energia
44	11	(0, 2)	contraste
45	11	(0, 2)	correlação
46	11	(0, 2)	variância
47	11	(0, 2)	mom. dif. inv.
48	11	(0, 2)	entropia
49	15	(1, 0)	energia
50	15	(1, 0)	contraste
51	15	(1, 0)	correlação
52	15	(1, 0)	variância
53	15	(1, 0)	mom. dif. inv.
54	15	(1, 0)	entropia
55	15	(0, 1)	energia
56	15	(0, 1)	contraste
57	15	(0, 1)	correlação
58	15	(0, 1)	variância
59	15	(0, 1)	mom. dif. inv.
60	15	(0, 1)	entropia
61	15	(2, 0)	energia
62	15	(2, 0)	contraste
63	15	(2, 0)	correlação
64	15	(2, 0)	variância
65	15	(2, 0)	mom. dif. inv.
66	15	(2, 0)	entropia
67	15	(0, 2)	energia
68	15	(0, 2)	contraste
69	15	(0, 2)	correlação
70	15	(0, 2)	variância
71	15	(0, 2)	mom. dif. inv.
72	15	(0, 2)	entropia

Tabela Apêndice A.7: Características extraídas das imagens médicas com matrizes de co-ocorrência.

<b>Caract.</b>	$\theta$	$f$
73	0°	0.050
74	90°	0.050
75	0°	0.055
76	90°	0.055
77	0°	0.061
78	90°	0.061
79	0°	0.068
80	90°	0.068
81	0°	0.075
82	90°	0.075
83	0°	0.083
84	90°	0.083
85	0°	0.092
86	90°	0.092
87	0°	0.102
88	90°	0.102
89	0°	0.113
90	90°	0.113
91	0°	0.126
92	90°	0.126

Tabela Apêndice A.8: Características extraídas das imagens médicas com filtros de Gabor.

<b>Caract.</b>	$\theta$	$f$
93	0°	0.139
94	90°	0.139
95	0°	0.154
96	90°	0.154
97	0°	0.171
98	90°	0.171
99	0°	0.189
100	90°	0.189
101	0°	0.210
102	90°	0.210
103	0°	0.232
104	90°	0.232
105	0°	0.257
106	90°	0.257
107	0°	0.285
108	90°	0.285
109	0°	0.316
110	90°	0.316
111	0°	0.350
112	90°	0.350

<b>Caract.</b>	$h$	<b>Função</b>
113	7	média
114	7	variância
115	7	energia
116	7	entropia
117	7	obliquidade
118	7	curtose
119	11	média
120	11	variância
121	11	energia
122	11	entropia
123	11	obliquidade
124	11	curtose
125	15	média
126	15	variância
127	15	energia
128	15	entropia
129	15	obliquidade
130	15	curtose

Tabela Apêndice A.9: Características extraídas das imagens médicas com estatísticas de primeira ordem.