

# Classificação Supervisionada de Padrões Utilizando Floresta de Caminhos Ótimos

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por João Paulo Papa e aprovada pela Banca Examinadora.

Campinas, 27 de novembro de 2008.

  
Alexandre Xavier Falcão (Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA  
BIBLIOTECA DO IMECC DA UNICAMP  
Bibliotecária: Maria Júlia Milani Rodrigues CRB8a / 2116**

Papa, João Paulo

P197c            Classificação supervisionada de padrões utilizando floresta de caminhos ótimos / João Paulo Papa -- Campinas, [S.P. :s.n.], 2008.

Orientador : Alexandre Xavier Falcão

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Reconhecimento de padrões. 2. Processamento de imagens. 3. Inteligência artificial. I. Falcão, Alexandre Xavier. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Supervised pattern classification using optimum path forest.

Palavras-chave em inglês (Keywords): 1. Pattern recognition. 2. Image processing. 3. Artificial intelligence.

Área de concentração: Metodologia e Técnicas da Computação

Titulação: Doutor em Ciência da Computação

Banca examinadora:

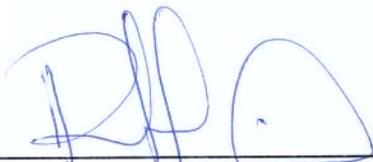
Prof. Dr. Alexandre Xavier Falcão (IC-UNICAMP)  
Prof. Dr. Roberto Hirata Júnior (IME-USP)  
Profa. Dra. Leila Maria Garcia Fonseca (DPI-INPE)  
Prof. Dr. Hélio Pedrini (IC-UNICAMP)  
Prof. Dr. Jacques Wainer (IC-UNICAMP)

Data da defesa: 27/11/2008

Programa de pós-graduação: Doutorado em Ciência da Computação

## TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 27 de novembro de 2008, pela Banca examinadora composta pelos Professores Doutores:



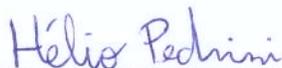
---

Prof. Dr. Roberto Hirata Júnior  
IME / USP.



---

Profª. Drª. Leila Maria Garcia Fonseca  
DPI / INPE.



---

Prof. Dr. Hélio Pedrini  
IC / UNICAMP.



---

Prof. Dr. Jacques Wainer  
IC / UNICAMP.



---

Prof. Dr. Alexandre Xavier Falcão  
IC / UNICAMP.

# Classificação Supervisionada de Padrões Utilizando Floresta de Caminhos Ótimos

João Paulo Papa<sup>1</sup>

Janeiro de 2009

## Banca Examinadora:

- Alexandre Xavier Falcão (Orientador)
- Roberto Hirata Júnior (IME-USP)
- Leila Maria Garcia Fonseca (DPI-INPE)
- Hélio Pedrini (IC-UNICAMP)
- Jacques Wainer (IC-UNICAMP)
- Aparecido Nilceu Marana (FC-UNESP) (suplente)
- Neucimar Jerônimo Leite (IC-UNICAMP) (suplente)
- Ricardo da Silva Torres (IC-UNICAMP) (suplente)

---

<sup>1</sup>Bolsista PED-A (IC/UNICAMP)

# Resumo

Padrões são geralmente representados por vetores de atributos obtidos através de amostras em uma base de dados, a qual pode estar totalmente, parcialmente ou não rotulada. Dependendo da quantidade de informação disponível dessa base de dados, podemos aplicar três tipos de técnicas para identificação desses padrões: supervisionadas, semi-supervisionadas ou não-supervisionadas. No presente trabalho, estudamos técnicas supervisionadas, as quais caracterizam-se pelo total conhecimento dos rótulos das amostras da base de dados. Propusemos também um novo método para classificação supervisionada de padrões baseada em Floresta de Caminhos Ótimos (OPF - Optimum-Path Forest), a qual modela o problema de reconhecimento de padrões como sendo um grafo, onde os nós são as amostras e os arcos definidos por uma relação de adjacência. Amostras mais relevantes (protótipos) são identificadas e um processo de competição entre elas é iniciado, as quais tentam oferecer caminhos de custo ótimo para as demais amostras da base de dados. Apresentamos aqui duas abordagens, as quais diferem na relação de adjacência, função de custo de caminho e maneira de identificar os protótipos. A primeira delas utiliza como relação de adjacência o grafo completo e identifica os protótipos nas regiões de fronteira entre as classes, os quais oferecem caminhos de custo ótimo que são computados como sendo o valor do maior peso de arco do caminho entre esses protótipos e as demais amostras da base de dados, sendo o peso do arco entre duas amostras dado pela distância entre seus vetores de características. O algoritmo OPF tenta minimizar esses custos para todas as amostras. A outra abordagem utiliza como relação de adjacência um grafo  $k$ -nn e identifica os protótipos como sendo os máximos de uma função de densidade de probabilidade, a qual é computada utilizando os pesos dos arcos. O valor do custo do caminho é dado pelo menor valor de densidade ao longo do caminho. Neste caso, o algoritmo OPF tenta agora maximizar esses custos. Apresentamos também um algoritmo de aprendizado genérico, o qual ensina o classificador através de seus erros em um conjunto de validação, trocando amostras classificadas incorretamente por outras selecionadas através de certas restrições. Esse processo é repetido até um critério de erro ser estabelecido. Comparações com os classificadores SVM, ANN-MLP,  $k$ -NN e BC foram feitas, tendo o OPF demonstrado ser similar ao SVM, porém bem mais rápido, e superior aos restantes.

# Abstract

Patterns are usually represented by feature vectors obtained from samples of a dataset, which can be fully, partially or non labeled. Depending on the amount of available information of these datasets, three kinds of pattern identification techniques can be applied: supervised, semi-supervised or non supervised. In this work, we addressed the supervised ones, which are characterized by the fully knowledge of the labels from the dataset samples, and we also proposed a novel idea for supervised pattern recognition based on Optimum-Path Forest (OPF), which models the pattern recognition problem as a graph, where the nodes are the samples and the arcs are defined by some adjacency relation. The most relevant samples (prototypes) are identified and a competition process between them is started, which try to offer optimum-path costs to the remaining dataset samples. We presented here two approaches, which differ from each other in the adjacency relation, path-cost function and the prototypes identification procedure. The first ones uses as the adjacency relation the complete graph and identify the prototypes in the boundaries of the classes, which offer optimum-path costs that are computed as been the maximum path arc-weight between these prototypes and the other dataset samples, in which the arc-weight is given by the distance between their feature vectors. In this case, the OPF algorithm tries to minimize these costs for each sample of the dataset. The other approach uses as the adjacency relation a  $k$ -nn graph and identifies the prototypes as the maxima of a probability density function, which is computed using the arc-weights. The path-cost value is given by the lowest density value among it. The OPF algorithm now tries to maximize these costs. We also presented a generic learning algorithm, which tries to teach a classifier through its errors in a validation set, replacing the misclassified samples by other selected using some constraints. This process is repeated until an error criterion is satisfied. Comparisons with SVM, ANN-MLP,  $k$ -NN and BC classifiers were also performed, being the OPF similar to SVM, but much faster, and superior to the remaining classifiers.

*Ao meu pai, Mauro Luiz Papa (in memoriam).*

## Súplicas de um jovem triste

Obrigado pelo cavalinho, papai  
Que me deste de presente  
Obrigado papai por este presentinho  
Que deixou-me muito contente

Obrigado pela linda bola  
Que eu sempre sonhei  
Com a rapaziada da escola  
Muito gabola fiquei  
Olhe ...  
Veja, quantos presentes ganhei  
Tenho de tudo, quanto quis  
Mas ...  
Falta-me tudo, quanto quis  
Mas ...  
Falta-me algo, que eu não sei  
Isso me aborrece.  
Sinto-me infeliz

Novos presentes, fico radiante  
Uma alegria que já considero efêmera  
Sinto um vazio  
Preciso de algo mais  
Que preencha esta lacuna  
Que corroe em meu âmago  
Novos presentes  
Meu pai me dá de “TUDO”  
Sinto-me ausente  
Meu coração bate valente  
Meu coração ficou mudo  
Como num baque surdo  
Sinto-me ainda mais distante

Hoje ganhei um carro

Fumo ...  
Bebo ...  
Não percebo  
O tempo passar  
Novamente envolto  
Em angustiante solidão  
Tudo para mim foi em vão  
Sinto-me infeliz novamente

Obrigado papai.

**Mauro Luiz Papa**

# Agradecimentos

A Deus, por iluminar meus passos e por sempre manter-me no caminho correto.

Em especial ao meu orientador, Prof. Alexandre Falcão, pelos conselhos infundáveis de quem já trilhou esse caminho, pela sua atenção, praticidade, preocupação, orientação, alegria e, principalmente, pela amizade adquirida.

Ao meu pai Mauro Luiz Papa (*in memoriam*) que, com certeza, está olhando por mim e minha família neste momento. **ESSA É PARA VOCÊ PAI!**

A toda a minha família, em especial a minha mãe Maria de Lourdes Sasso Papa, minhas irmãs Valéria, Hélen, Patrici e Helena, meus sobrinhos Gustavo e Bárbara e meus cunhados.

A minha namorada Greice Martins de Freitas que sempre esteve ao meu lado nestes últimos anos. Nunca esquecerei a sua ajuda nos momentos mais importantes, tanto os felizes quanto os tristes. Nunca esquecerei o seu carinho e paciência. Nunca esquecerei o seu olhar de criança ... **TE AMO MINHA MENINONA**, você foi a melhor coisa que aconteceu na minha vida ... Você me ensinou o que é amar ...

Aos meus amigos, tanto os de infância quanto os, não menos importantes, adquiridos durante a minha vida acadêmica. Aos colegas do Laboratório de Informática Visual (LIV), agradeço a simpatia e as valiosas trocas de informação e dicas.

Aos amigos Jancarlo e Celso, pelas constantes palavras de apoio, sempre necessárias.

À Universidade Estadual de Campinas (UNICAMP) por permitir tornar possível a realização do meu trabalho e aos professores e funcionários do Instituto de Computação, pelo apoio constante e informação adquiridos durante os meus estudos.

A todos que colaboraram, direta ou indiretamente, com o meu trabalho, e que estarei sempre em falta.

Que Deus nos abençoe, e que sempre nos permita ajudar os nossos semelhantes com a humildade necessária para tal tarefa.

*Podem tirar-lhe a liberdade e a dignidade. Podem até tirar-lhe a esperança. Mas nunca a fé e o conhecimento.*

# Sumário

<b>Resumo</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>Agradecimentos</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Revisão Bibliográfica</b>	<b>5</b>
2.1 Classificação não-supervisionada . . . . .	5
2.2 Classificação supervisionada . . . . .	6
2.3 Classificação semi-supervisionada . . . . .	9
<b>3 Transformada Imagem Floresta</b>	<b>10</b>
3.1 Definição . . . . .	10
3.2 Algoritmo da IFT . . . . .	12
3.3 Aplicações . . . . .	14
<b>4 Classificadores baseados em floresta de Caminhos ótimos</b>	<b>16</b>
4.1 Classificação não-supervisionada . . . . .	16
4.1.1 Fundamentação teórica . . . . .	17
4.1.2 Extensão para grandes bases de dados . . . . .	21
4.2 Classificação supervisionada . . . . .	22
4.2.1 Usando grafo completo . . . . .	22
4.2.2 Usando grafo $k$ -nn . . . . .	27
4.2.3 Aprendizado . . . . .	32
<b>5 Resultados Experimentais</b>	<b>35</b>
5.1 Bases de dados . . . . .	35
5.2 Descritores . . . . .	36

5.3	Resultados obtidos diretamente no conjunto de teste . . . . .	40
5.4	Resultados obtidos no conjunto de teste após aprendizado com conjunto de avaliação . . . . .	41
5.5	Resultados em eficiência . . . . .	42
<b>6</b>	<b>Conclusão e trabalhos futuros</b>	<b>46</b>
	<b>Bibliografia</b>	<b>52</b>

# Lista de Tabelas

5.1	Descrição das bases de dados. . . . .	36
5.2	Descritores utilizados nos experimentos. . . . .	39
5.3	Bases de dados e seus respectivos descritores utilizados nos experimentos. . . . .	40
5.4	Resultados $x \pm y(z)$ em $Z_3$ sem a utilização de $Z_2$ entre os algoritmos similares: $x$ - acurácia média, $y$ - desvio padrão das acurácias e $z$ - Kappa médio. As melhores acurácias são evidenciadas em negrito. . . . .	41
5.5	Resultados $x \pm y(z)$ em $Z_3$ sem a utilização de $Z_2$ entre os algoritmos populares: $x$ - acurácia média, $y$ - desvio padrão das acurácias e $z$ - Kappa médio. As melhores acurácias são evidenciadas em negrito. . . . .	42
5.6	Resultados $x \pm y(z)$ em $Z_3$ com aprendizado em $Z_2$ entre os algoritmos similares: $x$ - acurácia média, $y$ - desvio padrão das acurácias e $z$ - Kappa médio. As melhores acurácias são evidenciadas em negrito. . . . .	43
5.7	Resultados $x \pm y(z)$ em $Z_3$ com aprendizado em $Z_2$ entre os algoritmos populares: $x$ - acurácia média, $y$ - desvio padrão das acurácias e $z$ - Kappa médio. As melhores acurácias são evidenciadas em negrito. . . . .	44
5.8	Tempo de execução médio em segundos para os processos de treinamento e classificação dividido pelo número de amostras. . . . .	45

# Lista de Figuras

3.1	(a)-(c) Pixel central e seus 4-vizinhos, 8-vizinhos e uma relação de adjacência mais complexa, respectivamente. . . . .	11
3.2	(a) Um grafo de uma imagem 2D em tons de cinza com vizinhança 4. Os números correspondem às intensidades $I(s)$ dos pixels e os pontos maiores denotam as três sementes. (b) Uma floresta de caminhos ótimos usando $f_{max}$ com $d(s, t) = I(t)$ . As setas em (b) apontam para o predecessor no caminho ótimo. . . . .	12
3.3	Exemplo de segmentação utilizando a IFT. (a) Uma imagem de ressonância magnética do cérebro. (b) Imagem de gradiente de (a) com uma semente selecionada dentro do núcleo caudado (1) e outra fora (2). (c) Imagem segmentada resultante. . . . .	15
4.1	(a) Grafo cujos pesos dos nós são seus valores de fdp $\rho(t)$ . Existem dois máximos com valores 3 e 5, respectivamente. Os pontos grandes indicam o conjunto de raízes $S$ . (b) Valores de caminho triviais $f_1(\langle t \rangle)$ para cada amostra $t$ . (c) Floresta de caminhos ótimos $P$ para $f_1$ e os valores de caminho finais $V(t)$ . O caminho ótimo $P^*(t)$ (linha tracejada) pode ser obtido percorrendo os predecessores $P(t)$ até a raiz $R(t)$ para cada amostra $t$ . . . . .	19
4.2	(a) Espaço de atributos com diferentes concentrações de amostras para cada cluster. Podemos identificar diferentes quantidades de clusters dependendo do valor de $k$ escolhido. Soluções interessantes são (b) quatro e (c) cinco clusters. . . . .	20

4.3	(a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) MST do grafo completo. (c) Protótipos escolhidos como sendo os elementos adjacentes de classes diferentes na MST (nós circutados). (d) Floresta de caminhos ótimos resultante para a função de valor de caminho $f_{max}$ e dois protótipos. Os identificadores $(x, y)$ acima dos nós são, respectivamente, o custo e o rótulo dos mesmos. A seta indica o nó predecessor no caminho ótimo. (e) Uma amostra de teste (triângulo) da classe 2 e suas conexões (linhas pontilhadas) com os nós do conjunto de treinamento. (f) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2 e o custo de classificação 0.4 são associados à amostra de teste. Note que, mesmo a amostra de teste estando mais próxima de um nó da classe 1, ela foi classificada como sendo da classe 2. . . . .	25
4.4	(a) Floresta de caminhos ótimos obtida através do Algoritmo 4, cujos elementos são ponderados nos nós com seus respectivos valores de densidade. Os indicadores $(x, y)$ acima dos nós são, respectivamente, o seu valor de densidade e o rótulo da classe a qual ele pertence. Os elementos circutados representam os máximos de cada classe. (b) Processo de classificação, onde um nó $t \in Z_3$ (triângulo) a ser classificado é inserido no grafo e conectado aos seus $k$ -vizinhos mais próximos ( $k = 3$ no exemplo), e o seu valor de densidade é calculado. (c) Processo final da classificação, no qual a amostra $t$ é classificada com o rótulo da amostra $s \in Z_1$ que satisfaz a Equação 4.10, ou seja, $L(t) = L(s)$ . No exemplo acima, o elemento a ser classificado foi conquistado pelo máximo da classe 1. . . . .	30
4.5	Processo de classificação de uma amostra $t$ . (a) Classificador baseado em OPF com grafo $k$ -nn conecta a amostra $t$ aos seus $k$ -vizinhos para obter $P(t)$ . (b) BC conecta $t$ aos $k$ -vizinhos da classe $w_1$ (linha sólida) para obter $P(t w_1)$ e depois conecta $t$ aos $k$ -vizinhos da classe $w_2$ (linha pontilhada) para obter $P(t w_2)$ . Em seguida, conecta $t$ aos $k$ -vizinhos mais próximos (linha tracejada) para obter $P(t)$ e, conseqüentemente, usar a Regra de Bayes para classificar a amostra $t$ . . . . .	31
5.1	Exemplos de formas da base MPEG-7 das classes (a)-(c) peixes e (d)-(f) camelos. . . . .	36
5.2	Exemplos de imagens da base Corel das classes (a)-(b) abóbora, (c)-(d) flores, (e)-(f) exército britânico, e (g)-(h) carros de corrida. . . . .	37

5.3	Imagens de textura da base Brodatz. Cada imagem, da esquerda para a direita e de cima para baixo, representa uma classe: casca de árvore, tijolo, bolhas, grama, couro, pele de porco, fibra vegetal, areia, palha, água, tecido, madeira e lã. . . . .	37
5.4	Bases de dados de pontos 2D: (a) Cone-torus, (b) Saturno, (c) Pétalas, e (d) Boat. . . . .	38
5.5	Resultados obtidos diretamente no conjunto de teste para os classificadores $OPF_{cpl}$ , $OPF_{knn}$ , SVM, ANN-MLP, $k$ -NN e BC. . . . .	43
5.6	Curvas de aprendizado para os classificadores $OPF_{cpl}$ , $OPF_{k-nn}$ , SVM, ANN-MLP e $k$ -NN, usando o Algoritmo 6 com $I = 10$ iterações. Os valores de acurácia foram obtidos sobre o conjunto avaliação $Z_2$ na base de dados Cone-torus. . . . .	44
5.7	Resultados obtidos com aprendizado em $Z_2$ para os classificadores $OPF_{cpl}$ , $OPF_{knn}$ , SVM, ANN-MLP, $k$ -NN e BC. . . . .	45

# Capítulo 1

## Introdução

Dado um conjunto de amostras (*pixels*, *voxels*, contornos, regiões, imagens), que pode ser particionado em  $c$  classes (rótulos), a classificação de padrões visa encontrar a classe de cada amostra usando um vetor de atributos. Os métodos normalmente aprendem como as amostras se distribuem em diferentes classes no espaço de atributos, usando um subconjunto de treinamento e encontrando superfícies ou regras de decisão para o particionamento deste conjunto [22].

A classificação automática tem sido um grande desafio em uma variedade de problemas científicos e comerciais. Métodos de aprendizado podem ser divididos em supervisionados, não-supervisionados, e semi-supervisionados, de acordo com seus algoritmos de aprendizado [39]. Métodos não-supervisionados não possuem ou não exploram conhecimento algum sobre as classes das amostras de treinamento, enquanto esta informação é conhecida e explorada em métodos supervisionados. Métodos semi-supervisionados conhecem apenas parte ou fazem uso parcial dos rótulos das amostras de treinamento.

Em métodos não-supervisionados, o aprendizado visa agrupar amostras similares como sendo de uma mesma classe, mas não existe garantia que estas amostras serão associadas à classe correta. As técnicas se dividem entre aquelas que modelam o problema usando grafos [74, 41, 40] e aquelas que não fazem uso dos conceitos de grafos, como o famoso algoritmo  $k$ -médias [49]. Métodos semi-supervisionados associam um rótulo a cada amostra de treinamento com base no conhecimento dos rótulos de parte delas [5, 8, 43]. Estes métodos também não garantem que as amostras de treinamento serão classificados corretamente. Na verdade, nem a maioria das abordagens supervisionadas garante erro de classificação zero no conjunto de treinamento.

Entre as abordagens supervisionadas, Redes Neurais Artificiais (*Artificial Neural Networks* - ANN) [36] e Máquinas de Vetores de Suporte (*Support Vector Machines* - SVM) [9] são as mais populares. O modelo mais usado de ANN usa Perceptrons em Múltiplas Camadas (*Multi-Layer Perceptrons* - MLP), assumindo que as classes são linearmente

separáveis por partes. ANN-MLP é um modelo instável (o desempenho oscila muito dependendo do conjunto de treinamento) e seu desempenho pode melhorar quando combinamos várias redes [46], gerando um maior custo computacional. Mesmo assim, o aumento no número de redes pode também provocar piora de acurácia e o número ideal de redes é desconhecido [62]. Como a premissa de separabilidade linear por partes no espaço de atributos não é sempre válida, SVM busca uma separabilidade linear ótima em um espaço de maior dimensão. Como desvantagens, esta premissa pode não ser válida com dimensão finita [17] e o custo computacional do SVM aumenta muito rápido com o número de vetores de suporte [68, 55].

Técnicas de reconhecimento estatístico de padrões tentam determinar superfícies de decisão baseadas em distribuições de probabilidade que melhor representam cada classe [42], as quais podem ser conhecidas a priori ou necessitam ser estimadas. Se os rótulos forem conhecidos, então um classificador Bayesiano (*Bayesian Classifier* - BC) pode ser utilizado para a tarefa de classificação das amostras. Em situações nas quais não temos informação alguma sobre a densidade das classes, temos um problema de decisão não paramétrico, e tais densidades deverão ser estimadas.

Apesar da abordagem supervisionada ser a que melhor possibilita o projeto de um classificador robusto, por explorar mais informação *a priori*, a literatura ainda carece de um método capaz de tratar classes não-separáveis sem assumir forma ou modelo paramétrico para essas classes. Mais ainda, a conectividade entre pixels de uma imagem tem sido explorada no contexto de segmentação de imagens, no espaço de atributos a conectividade foi explorada apenas localmente através de distâncias diretas entre amostras. Nós investigamos, neste trabalho, a importância da conectividade global, a qual leva em conta uma cadeia de amostras fortemente conexas no espaço de atributos para fins de classificação. Este trabalho visa, portanto, preencher esta lacuna com a proposta de um método para projeto de classificadores supervisionados baseados em florestas de caminhos ótimos (OPF-*Optimum Path Forest*).

Classificadores baseados em OPF baseiam-se em um subconjunto de amostras *protótipos* (i.e., representantes de todas as classes do conjunto de treinamento) e em uma função de custo de caminho no grafo de treinamento, a qual busca agrupar amostras com atributos similares. Tais classificadores particionam o grafo de treinamento em uma floresta de caminhos de custo mínimo, onde cada árvore é enraizada em um protótipo e todas as amostras da árvore são rotuladas com o mesmo rótulo da raiz. A classificação de uma nova amostra se dá por encontrar o protótipo que lhe oferece o caminho ótimo entre os caminhos oferecidos por todos os protótipos, ou seja, a classificação é baseada na força de conexão entre a amostra e o protótipo mais conexo, e não na simples distância entre eles, como faz, por exemplo, os classificadores baseados em *k*-vizinhos mais próximos [34].

Neste trabalho exploramos duas abordagens supervisionadas, sendo que a primeira

delas modela o problema como sendo um grafo completo, onde o peso da aresta entre duas amostras é dado pela distância entre elas. Para esta metodologia, utilizamos a função  $f_{\max}$ , a qual associa como custo de um caminho o valor do arco de maior peso ao longo dele. Esta abordagem trata o problema de classes não-separáveis estimando protótipos nas regiões de intersecção entre as classes.

A outra metodologia, também baseada em floresta de caminhos ótimos, usa uma função de densidade de probabilidade (fdp) para ponderar os nós de um grafo incompleto, cujos arcos são definidos por uma relação de adjacência a qual, para cada nó, considera os seus  $k$ -vizinhos mais próximos (grafo  $k$ -nn). Os pesos dos arcos são, novamente, as distâncias entre as amostras. Já os nós são ponderados pelos seus valores de densidade de probabilidade, os quais são computados usando os pesos dos arcos. A função de valor de caminho associa ao nó terminal  $s$  de cada caminho o mínimo entre os valores de densidade ao longo do mesmo e um valor de densidade inicial. Neste caso, a floresta de caminhos ótimos é obtida maximizando os valores das densidades ao longo de todos os caminhos entre o conjunto de protótipos e as amostras. Essa função é essencialmente dual de  $f_{\max}$ , ou seja, é uma função  $f_{\min}$  baseado nos pesos dos vértices. As raízes da floresta (protótipos) formam um subconjunto dos máximos da fdp onde, cada raiz, define uma árvore de caminhos ótimos (zona de influência do respectivo máximo) composta pelas suas amostras mais fortemente conexas. Esta abordagem difere de classificadores bayesianos, por exemplo, por explorar a conectividade entre as amostras do conjunto de dados.

Outra contribuição deste trabalho está relacionada com algoritmos de aprendizagem, os quais podem ensinar um classificador a aprender com seus erros em um terceiro conjunto, avaliação, sem aumentar o tamanho do conjunto de treinamento. Como as amostras do conjunto de teste geralmente não são vistas durante o projeto de um classificador, um novo conjunto de avaliação se faz necessário para este tipo de aprendizado. Basicamente, a idéia é trocar aleatoriamente amostras do conjunto de treinamento com amostras classificadas incorretamente no conjunto de avaliação, re-treinar o classificador e avaliá-lo novamente, repetindo este procedimento durante algumas iterações. É esperada uma melhora no desempenho do classificador.

A motivação para o aprendizado com os próprios erros sem aumentar o tamanho do conjunto de treinamento vem de aplicações onde o conjunto de dados (*voxels* de imagens tomográficas, por exemplo) é muito grande e nós precisamos reduzir o conjunto de treinamento, mas garantindo as amostras mais relevantes para o treinamento. Existem também aplicações onde o classificador faz parte de um sistema especialista, o qual faz análise de dados laboratoriais e emite sua opinião para um técnico responsável. Este especialista pode aceitar ou não o diagnóstico dado pelo sistema. Em caso de não aceitação, o profissional pode alimentar o sistema indicando seus erros de classificação, para uma possível melhora no desempenho do sistema no futuro. Um exemplo real seria o diagnóstico de

parasitos em imagens de microscopia. A inspeção visual humana é muito difícil e o erro no diagnóstico em várias situações devido à quantidade de impurezas e o pequeno tamanho de alguns parasitos (protozoários em amostras fecais, por exemplo) é muito alto. Nós desejamos melhorar o desempenho de sistemas especialistas ao longo do tempo, levando também em consideração a capacidade limitada de armazenamento dos computadores.

O algoritmo de floresta de caminhos ótimos é o mesmo da Transformada Imagem Floresta (IFT - *Image Foresting Transform* [27]), a qual tem sido utilizada com sucesso em várias aplicações de processamento de imagens [70, 23, 26]. O presente trabalho essencialmente estende as aplicações da IFT (domínio da imagem) para classificação de padrões (espaço de atributos).

Esta tese de doutorado está dividida da seguinte forma: no Capítulo 2 apresentamos uma revisão das principais técnicas de reconhecimento de padrões. Os Capítulos 3 e 4 apresentam, respectivamente, a IFT e sua extensão ao espaço de atributos, e os classificadores baseados em OPF. Os resultados comparativos com ANN-MPL, SVM,  $k$ -NN e BC são apresentados no Capítulo 5. Finalmente, as conclusões e trabalhos futuros são apresentados no Capítulo 6.

# Capítulo 2

## Revisão Bibliográfica

### 2.1 Classificação não-supervisionada

Em muitas aplicações de reconhecimento de padrões, rotular corretamente uma amostra de treinamento é uma tarefa extremamente difícil e custosa, ou até mesmo impossível. Isto deve-se principalmente a vários fatores, tais como sobreposição de classes e ao fato de em grande parte dos casos nenhuma informação *a priori* da distribuição dos dados estar disponível. Métodos de classificação não-supervisionados referem-se a situações onde o objetivo principal é construir limiares de decisão baseados no conjunto de dados não rotulado, ou seja, agrupar amostras que compartilham certas “semelhanças”. Esta noção de similaridade pode ser expressa de várias formas, de acordo com o propósito do estudo. Amostras pertencentes a uma dada classe irão formar um agrupamento que irá se caracterizar pela alta densidade de amostras em uma dada região do espaço de atributos.

Entretanto, a classificação não-supervisionada ou *clustering* (agrupamento) é um problema de difícil tratamento. As amostras podem compor *clusters* (grupos) com diferentes tamanhos e formas. O número de tais agrupamentos pode variar sensivelmente, dependendo da resolução na qual os dados são trabalhados [22]. Tais atributos tornam as metodologias baseadas nessas abordagens muito sensíveis ao conjunto de descritores utilizados para representar as amostras. Como consequência, muitos algoritmos têm sido propostos com o intuito de resolver tais limitações. Tais abordagens são basicamente baseadas em duas técnicas muito populares: clustering hierárquico e clustering particional. Algoritmos baseados na metodologia hierárquica organizam os dados em uma seqüência de grupos, os quais podem ser visualizados na forma de um dendrograma ou árvore. Métodos particionais tentam obter uma partição que maximiza a similaridade entre amostras de um mesmo grupo. O problema pode ser formulado da seguinte maneira: dadas  $n$  amostras em um espaço  $N$  dimensional, a idéia é encontrar uma partição das mesmas tais que elementos pertencentes a um mesmo grupo são mais similares do que el-

elementos pertencentes a grupos distintos. Para garantir a obtenção de uma solução ótima, uma abordagem seria examinar todas as possíveis partições, o que não é uma tarefa computacionalmente viável. Assim sendo, várias heurísticas têm sido utilizadas para reduzir a busca, porém sem a garantia de encontrar uma solução ótima [42].

Técnicas de classificação não-supervisionadas também são comumente associadas a métodos baseados em grafos. Zahn [74] propôs uma abordagem que computa uma árvore geradora mínima (*Minimum Spanning Tree* - MST) no grafo de treinamento e remove as arestas inconsistentes (arestas que separam agrupamentos diferentes), com o intuito de formar os grupos. Existem várias abordagens para identificar tais arestas, como por exemplo remover um arco cujo peso é significativamente maior do que a média dos pesos dos arcos vizinhos; ou ainda atribuir um fator de inconsistência a um arco, que pode ser calculado pela razão entre o peso de um arco e a média dos pesos dos arcos mais próximos. Grafos de vizinhança, tais como o grafo de vizinhança relativa (*Relative Neighborhood Graph* - RNG) e o Grafo de Gabriel (*Gabriel Graph* - GG), também têm sido utilizados na análise de clusters, os quais são baseados em regiões de influência [41]. Duas amostras são consideradas vizinhas em um RNG caso nenhuma outra amostra pertença à intersecção das regiões de influência delas, as quais são definidas como sendo discos de raio  $d$  (distância entre elas) centralizados nessas amostras. O GG é definido semelhantemente, porém a região de influência de cada amostra é dada por um disco de diâmetro  $d$ . Tais abordagens possuem como principal deficiência levar em consideração apenas a proximidade entre as amostras, funcionando bem apenas em situações cujos grupos são disjuntos, o que dificilmente ocorre na prática.

Outras técnicas também utilizam remoção de arcos através da MST, as quais produzem soluções hierárquicas, tais como o popular single-linkage [40]. Esta abordagem utiliza os chamados grafos limiares (*Threshold Graphs*), onde duas amostras são conectadas caso a distância entre ambas seja menor que um dado limiar. Entretanto, este critério não é adequado em situações mais complexas. Técnicas de agrupamento têm sido também formuladas como problemas de corte em grafo [73, 64] com aplicações em segmentação de imagens, onde o grafo não precisa ser completo.

Essencialmente, métodos de clustering baseados em grafo têm como principal objetivo particionar o grafo em componentes, ou seja, grupos. Contudo, não há garantia que amostras em um dado grupo pertençam à mesma classe, sendo uma tarefa difícil associar tais elementos à sua classe correta sem nenhum conhecimento *a priori* dos mesmos.

## 2.2 Classificação supervisionada

Metodologias baseadas em classificação supervisionada caracterizam-se pelo conhecimento *a priori* do rótulo de todo o conjunto de dados de treinamento. Entretanto, muitas

abordagens encontradas na literatura ainda não conseguem fazer o mapeamento correto das classes no conjunto de treinamento.

São inúmeras as técnicas de classificação supervisionada de padrões. Dentre as mais conhecidas, podemos citar redes neurais artificiais (ANN), as quais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Para o seu treinamento, o algoritmo de retropropagação é o mais utilizado. O principal objetivo deste algoritmo é desenvolver uma regra de treinamento com o intuito de minimizar o erro quadrático total entre as saídas desejadas e as saídas reais dos nós em uma camada de saída.

Entretanto, uma ANN baseada em perceptron multicamadas (ANN *Multilayer Perceptron* ANN-MLP) treinada por retropropagação, por exemplo, é um classificador instável (sua acurácia varia muito com diferentes conjuntos de treinamento). Seu desempenho pode ser consideravelmente melhorada com o uso de múltiplos classificadores (*bagging - boosting*) [46]. Porém, à medida que se aumenta o número de classificadores, corre-se o risco de aumentar também a complexidade de todo o sistema, influenciando negativamente no seu desempenho, como mostram Reyzin e Schapire [62]. Aliado a isto tem-se o fato que uma ANN-MLP assume que as classes podem ser separadas por hiperplanos no espaço de atributos. Tal afirmativa muitas vezes não é válida na prática.

Outro método amplamente utilizado é o SVM, o qual propõe resolver o problema de classificação de padrões assumindo ser possível separar as classes em um espaço de mais alta dimensão. Suponha uma situação na qual os dados não são linearmente separáveis. Tais amostras podem ser dicotomizadas usando curvas ou círculos como superfícies de decisão, porém encontrar tais limiares é uma tarefa custosa. A principal idéia de uma SVM é pré-processar os dados de tal forma que o problema de encontrar uma função discriminante não linear seja transformado em um problema de encontrar um hiperplano, ou seja, mapear os dados que estão em uma dimensão qualquer para outra maior, tornando os mesmos linearmente separáveis. Isto é feito definindo um mapeamento o qual transforma o vetor de entrada em um outro (usualmente maior) vetor. Espera-se que, escolhendo um mapeamento adequado, o novo conjunto de treinamento seja linearmente separável.

Embora isso permita um bom desempenho, seu custo computacional aumenta rapidamente de acordo com o tamanho do conjunto de treinamento e o número de vetores de suporte [68]. Panda et al. [55] apresentaram um algoritmo que tem como principal objetivo diminuir o conjunto de treinamento, eliminando as amostras que não são vetores de suporte. Ainda que Tang e Mazzoni [68] tenham proposto um método de reduzir o número de vetores de suporte no caso de várias classes, o mesmo sofre pela demora na convergência e complexidade, visto que o trabalho propõe-se a resolver esse problema minimizando o número de vetores de suporte em várias SVMs binárias e depois compartilhando os sub-

conjuntos de vetores de suporte de cada uma dessas máquinas. Projetado inicialmente para ser um classificador binário, múltiplas SVMs são necessárias para resolver problemas com várias classes [21]. Duas abordagens principais são a um-contra-todos (*one-versus-all* - OVA) e um-contra-um (*one-versus-one* - OVO). OVA utiliza  $c$  SVMs para separar cada classe das outras. A decisão é dada para a classe com maior valor de confiança. Já a abordagem OVO requer  $\frac{c(c-1)}{2}$  SVMs, as quais levam em consideração todas as combinações binárias entre as classes. A decisão é dada, geralmente, pela maioria de votos. Devemos atentar também ao fato de a separabilidade proposta pela SVM pode não ser válida em um espaço de dimensão finita [47]. Assim sendo, ainda é desejável desenvolver um classificador que possa tratar classes não separáveis com um conjunto de treinamento de tamanho razoável.

Outras abordagens empregadas são as que utilizam algum conhecimento estatístico sobre o conjunto de dados, dependendo do tipo de informação disponível. Caso todas as densidades condicionais sejam conhecidas, então a regra de decisão de Bayes pode ser utilizada. Contudo, se as funções de densidade de probabilidade forem apresentadas, mas não seus parâmetros, tem-se um problema de decisão paramétrico. A principal metodologia abordada, neste caso, é a estimação dos parâmetros em questão [42]. Ainda assim, caso as funções de densidade de probabilidade das classes não sejam conhecidas, temos um problema de decisão não paramétrico, no qual tais funções de densidade deverão ser estimadas utilizando, por exemplo, janelas de Parzen ou  $k$ -vizinhos.

Outro método bastante utilizado é o conhecido  $k$ -vizinhos mais próximos ( $k$ -NN - *k-Nearest Neighbors*) [34] o qual, basicamente, classifica as amostras com base nos rótulos das amostras mais próximas do conjunto de treinamento. Na fase de treinamento, o espaço de atributos é particionado em regiões de acordo com os rótulos das amostras de treinamento. Uma amostra é associada à classe mais freqüente de suas  $k$ -vizinhas mais próximas na etapa de classificação. Apesar de ser uma abordagem simples de ser implementada, a acurácia do algoritmo  $k$ -NN pode ser severamente degradada pela presença de ruído (outliers) ou dados irrelevantes. O custo computacional aumenta com o valor de  $k$ , sendo sua escolha um grande problema. Desta forma, a melhor escolha de  $k$  depende intrinsecamente do seu conjunto de dados: altos valores de  $k$  tendem a amenizar o efeito do ruído, porém cria superfícies de separação que tornam as classes menos distintas. Geralmente o parâmetro  $k$  é escolhido através do método de otimização por validação cruzada (*cross validation*). Um caso particular do  $k$ -NN é quando a classe de uma amostra do conjunto de teste é dada pelo rótulo de seu vizinho mais próximo. Neste caso, temos o algoritmo de vizinho mais próximo (NN - *Nearest Neighbor*) [19].

## 2.3 Classificação semi-supervisionada

Como mencionado, no aprendizado supervisionado assumimos possuir um conjunto de amostras de treinamento rotulado, sendo a tarefa construir uma função que irá corretamente prever os rótulos das amostras futuras. No aprendizado não-supervisionado a idéia é segmentar as amostras não rotuladas em grupos, de tal forma a agrupar elementos similares e separar amostras que não possuam atributos em comum. Já na aprendizagem semi-supervisionada, assumimos ter tanto amostras rotuladas como não rotuladas no conjunto de treinamento. Desta forma, a tarefa é prever a classe dos elementos futuros utilizando esses dois tipos de amostras. Algoritmos semi-supervisionados que aprendem tanto através de amostras rotuladas como não rotuladas têm sido foco de muitos trabalhos de pesquisa [5, 8, 43]. Tais estudos assumem que pontos próximos e amostras no mesmo grupo possuem rótulos similares.

Basu et al. [5] apresentam um algoritmo semi-supervisionado o qual é uma modificação do já conhecido  $k$ -médias. Este algoritmo utiliza um subconjunto dos dados rotulados como sementes para iniciar e guiar um processo de agrupamento de dados. O inconveniente é que tal conjunto de dados inicial necessita ser selecionado pelo usuário.

Blum e Mitchell [8] e Joachims [43] apresentam, respectivamente, o método do co-treinamento (*Co-Training*) e a Máquina de Vetores de Suporte Transdutiva (*Transductive Support Vector Machine* - TSVM), a qual é uma extensão da tradicional SVM. No co-treinamento, as amostras são divididas em dois subconjuntos. Inicialmente, dois classificadores supervisionados são treinados em cada subconjunto, separadamente. Em seguida, cada classificador faz a sua predição no conjunto de dados não rotulado (amostras de teste), ensinando o outro classificador com as suas amostras classificadas. As amostras de teste mais confiantes de um classificador são utilizadas para aumentar o conjunto de treinamento do outro, e vice-versa. Assim sendo, o processo de retreinamento se repete. Porém o método assume que o subconjunto de cada classificador é suficientemente bom, de tal forma que podemos confiar em sua classificação.

Geralmente, no algoritmo da SVM, somente o conjunto de dados rotulado é utilizado no processo de treinamento, com o intuito de descobrir os vetores de suporte e estruturar os hiperplanos de separação ótimos. Entretanto, na TSVM, um conjunto não rotulado também é utilizado. Contudo, encontrar a solução exata em uma TSVM é um problema NP-difícil [76]. Outras limitações de tais métodos também podem ser elucidadas. Ghani [35] demonstrou que o uso de conjuntos de dados não rotulados com a TSVM pode causar uma degradação do seu desempenho. Já Zhang e Oles [75] descreveram experimentos que apontam o mesmo fenômeno no método de co-treinamento.

# Capítulo 3

## Transformada Imagem Floresta

A Transformada Imagem Floresta (IFT) é uma ferramenta geral para modelar, implementar e avaliar operadores de processamento de imagens baseados em conectividade [27]. A IFT reduz problemas de processamento da imagem ao cálculo de uma floresta de caminhos de custo ótimo em um grafo derivado da imagem. O valor de um caminho é normalmente calculado por uma função dependente da aplicação e com base em propriedades da imagem, tais como brilho, gradiente e posição do pixel ao longo do caminho.

### 3.1 Definição

Seja  $\hat{I} = (D_I, I)$  uma imagem 2D/3D, a qual pode ser vista como um grafo onde os nós são os pixels/voxels (amostras) e as arestas são definidas por uma relação de adjacência  $A$  entre nós (Figuras 3.1a-c mostram um pixel central e seus 4-vizinhos, 8-vizinhos e uma relação de adjacência mais complexa, respectivamente). Um caminho nesse grafo é uma seqüência de amostras  $\pi_{s_k} = \langle s_1, s_2, \dots, s_k \rangle$ , onde  $(s_i, s_{i+1}) \in A$  para  $1 \leq i \leq k - 1$ . Um caminho é dito ser trivial se  $\pi_s = \langle s \rangle$ . Nós associamos a cada caminho  $\pi_s$  um valor dado por uma função de valor de caminho  $f$ , denotada  $f(\pi_s)$ . Dizemos que um caminho  $\pi_s$  é ótimo se  $f(\pi_s) \leq f(\tau_s)$  para qualquer caminho  $\tau_s$ , onde  $\pi_s$  e  $\tau_s$  terminam na mesma amostra  $s$ , independente de sua origem. Também denotamos  $\pi_s \cdot \langle s, t \rangle$  a concatenação do caminho  $\pi_s$  com término em  $s$  e o arco  $(s, t)$ . O algoritmo da IFT pode ser utilizado com qualquer função de valor de caminho suave. Uma função de valor de caminho  $f$  é suave quando, para qualquer amostra  $t$ , existe um caminho ótimo  $\pi_t$  o qual é trivial ou possui a forma  $\pi_s \cdot \langle s, t \rangle$ , onde

- $f(\pi_s) \leq f(\pi_t)$ ;
- $\pi_s$  é ótimo, e

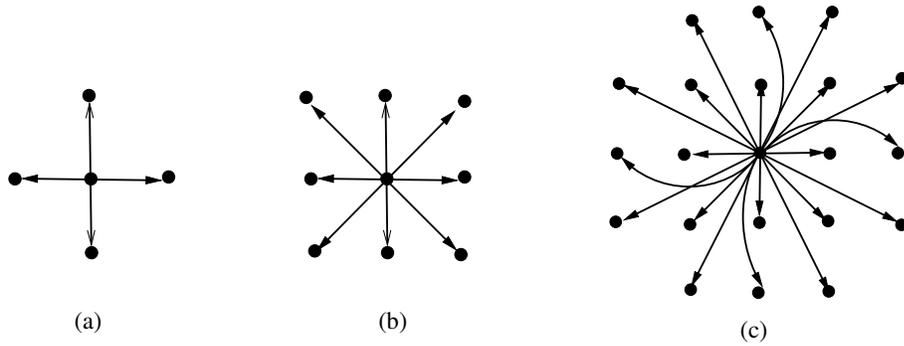


Figura 3.1: (a)-(c) Pixel central e seus 4-vizinhos, 8-vizinhos e uma relação de adjacência mais complexa, respectivamente.

- para qualquer caminho ótimo  $\tau_s$ ,  $f(\tau_s \cdot \langle s, t \rangle) = f(\pi_t)$ .

Em aplicações típicas da IFT, normalmente restringimos a busca por caminhos que se originam em um conjunto dado  $S$  de nós *sementes*. Tal conjunto  $S$  pode ser embutido na definição da função de valor de caminho. Como exemplo, podemos citar a função de valor de caminho  $f_{max}$ , a qual é definida como

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi_s \cdot \langle s, t \rangle) &= \max\{f_{max}(\pi_s), d(s, t)\}, \end{aligned} \quad (3.1)$$

sendo que  $d(s, t)$  mede a dissimilaridade entre nós adjacentes e  $f_{max}(\pi_s)$  computa a distância máxima entre amostras adjacentes em  $\pi_s$ , quando  $\pi_s$  não é um caminho trivial. Suponha, por exemplo, o grafo da Figura 3.2a, onde os pixels são os nós e as arestas são formadas pela 4-vizinhança (Figura 3.1a). Note que existem três sementes (nós de maior tamanho). Se utilizarmos a função  $f_{max}$  com  $d(s, t) = I(t)$ , onde  $I(t)$  denota o brilho do pixel  $t$ , a IFT encontra uma floresta de caminhos ótimos com raízes neste conjunto de sementes, como pode ser visualizado na Figura 3.2b. Neste caso, a IFT tenta minimizar os valores dos caminhos, os quais são dados pelo valor máximo de brilho ao longo dos mesmos ( $f_{max}$ ).

Outra função de valor de caminho amplamente utilizada é a  $f_{sum}$ , dada por

$$\begin{aligned} f_{sum}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{sum}(\pi_s \cdot \langle s, t \rangle) &= f_{sum}(\pi_s) + d(s, t), \end{aligned} \quad (3.2)$$

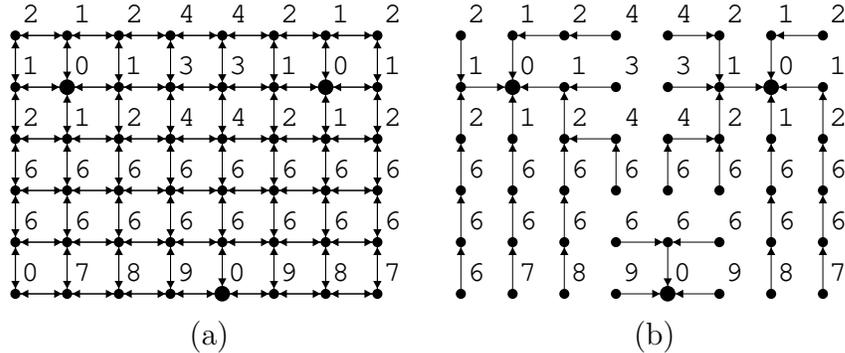


Figura 3.2: (a) Um grafo de uma imagem 2D em tons de cinza com vizinhança 4. Os números correspondem às intensidades  $I(s)$  dos pixels e os pontos maiores denotam as três sementes. (b) Uma floresta de caminhos ótimos usando  $f_{max}$  com  $d(s, t) = I(t)$ . As setas em (b) apontam para o predecessor no caminho ótimo.

a qual computa o somatório das distâncias ao longo do caminho. Neste caso, a IFT encontra os caminhos de menor distância acumulada a partir do conjunto de sementes para todas as outras amostras do grafo.

Para funções de valor de caminhos suaves, a IFT produz um caminho de valor ótimo indo do conjunto de sementes para cada nó do grafo, de tal maneira que a união destes caminhos forma uma floresta orientada, estendendo-se por toda a imagem. A IFT produz três atributos para cada pixel/voxel: seu predecessor no caminho ótimo, o valor deste caminho e o nó raiz correspondente (ou algum rótulo associado a ele). Uma grande variedade de operadores de imagem pode ser implementada através de simples processamento local destes atributos [26, 24, 70, 6].

## 3.2 Algoritmo da IFT

Um *mapa de predecessores*  $P$  é uma função que atribui para cada pixel  $s$  da imagem algum outro pixel ou uma marca distinta *nil* indicando a ausência de predecessor. Neste último caso,  $s$  é dito ser *raiz* do mapa. Dizemos também que  $P^*(s)$  denota o caminho ótimo da raiz  $R(s)$  até  $s$ . Uma floresta pode ser representada em memória através de um mapa de predecessores que não contém ciclos. O algoritmo da IFT retorna um mapa de predecessores  $P$  representando a floresta ótima, um mapa de valores de caminho  $V$  e um mapa de raízes  $R$ , o qual é utilizado para acessar em tempo constante a raiz em  $S$  de cada pixel da floresta. O mapa  $V$  armazena, para cada pixel, o valor do caminho ótimo que o alcança a partir do conjunto  $S$  de sementes mencionado.

O algoritmo da IFT abaixo é essencialmente o procedimento de Dijkstra para o cálculo de caminhos de valor ótimo a partir de uma única fonte [2, 20], ligeiramente modificado para permitir múltiplas fontes e funções de valor de caminho mais genéricas (funções suaves).

### Algoritmo 1 – IFT

ENTRADA: Uma imagem, uma relação de adjacência  $A$ , um conjunto de nós sementes  $S$  e uma função suave de valor de caminho  $f$ .

SAÍDA: Mapa de valores de caminhos  $V$ , mapa de predecessores  $P$  e mapa de raízes  $R$

AUXILIARES: Fila de prioridades  $Q$  inicialmente vazia e variável  $cst$ .

1. **Para cada** nó  $s$  do grafo derivado da imagem, **Faça**
2.      $P(s) \leftarrow nil, R(s) \leftarrow s$  e  $V(s) \leftarrow f(\langle p \rangle)$ .
3.     **Se**  $V(s)$  for finito, **Então**
4.          $\hookrightarrow$  **Insira**  $s$  em  $Q$ .
5. **Enquanto**  $Q$  não for vazia, **Faça**
6.      $\hookrightarrow$  **Remova**  $s$  de  $Q$  tal que  $V(s)$  é mínimo.
7.     **Para cada** nó  $t \in A(s)$  tal que  $V(t) > V(s)$ , **Faça**
8.          $\hookrightarrow$   $cst \leftarrow f(P^*(s) \cdot \langle s, t \rangle)$ .
9.         **Se**  $cst < V(t)$ , **Então**
10.              $\hookrightarrow$  **Se**  $V(t)$  for finito, **Então**
11.                  $\hookrightarrow$  **Remova**  $t$  de  $Q$ .
12.                  $\hookrightarrow$   $P(t) \leftarrow s, R(t) \leftarrow R(s)$  e  $V(t) \leftarrow cst$ .
13.                  $\hookrightarrow$  **Insira**  $t$  em  $Q$ .
14. **Retorne**  $\{V, P, R\}$

As linhas 1–4 inicializam a floresta como um conjunto de árvores triviais, nós isolados a serem conquistados durante o processo. Os custos são iniciados com  $f(\langle p \rangle)$  refletindo que nenhum caminho a partir das sementes foi processado. No caso da função  $f_{max}$  (Equação 3.1), por exemplo, temos que  $V(s) \leftarrow 0$  caso  $s \in S$  e  $V(s) \leftarrow +\infty$  caso contrário. Assim, os caminhos triviais a partir das sementes são inicializados. Tais caminhos possuem o valor mínimo 0 (para o caso da função  $f_{max}$ , por exemplo) de forma que todas as sementes se tornarão, obrigatoriamente, raízes da floresta. As sementes são inseridas na fila de prioridades  $Q$  (Linhas 3–4). Os pixels presentes na fila de prioridades representam a fronteira da floresta em crescimento, os quais correspondem a nós da floresta atingidos por caminhos não necessariamente ótimos. A cada iteração do algoritmo (linha 5) um caminho ótimo é selecionado, o qual corresponde ao caminho de menor valor entre os nós que atingem a fronteira da floresta (linha 6) e os seus vértices adjacentes são avaliados (linha 7). A fronteira pode ser ampliada pela aquisição de novas conexões ou melhores

rotas podem ser encontradas para pixels de fronteira já existentes. Na linha 8 é calculado o custo  $cost$  de uma nova possível rota, o qual é comparado com o valor do caminho atual (linha 9). Os mapas  $V$ ,  $P$  e  $R$  devem ser atualizados de forma a refletirem o melhor caminho encontrado (linha 12). A condição  $V(t) > V(s)$  na linha 7 é uma otimização que explora o fato de o valor ao longo do caminho ótimo não ser decrescente. Assim sendo, quando temos várias sementes em  $S$ , estas serão propagadas ao mesmo tempo e teremos um processo competitivo. Cada semente irá definir uma *zona de influência* composta por pixels conexos a ela por caminhos mais “baratos” do que os fornecidos por qualquer outra semente em  $S$ .

### 3.3 Aplicações

A IFT fornece um comum e eficiente *framework* baseado em grafos para o desenvolvimento de métodos de segmentação de imagens 2D/3D baseados em borda [33, 32, 31], bem como para métodos baseados em região [23, 48, 26, 12, 4, 6]. As aplicações deste algoritmo, na sua forma mais genérica [27], também incluem diversos operadores de filtragem e análise de imagens: caminhos geodésicos, transformada de distância [25], esqueletos multiescala [25], dimensão fractal multiescala [70], saliências de formas [69] reconstruções morfológicas e outras operações conexas [26].

Existem aplicações as quais desejam maximizar ao invés de minimizar os valores dos caminhos (Seção 4.1), tais como a identificação de máximos regionais, por exemplo, a qual pode ser definida por uma função  $f_{min}$ :

$$f_{min}(\langle s \rangle) = I(s) + 1 \text{ se } s \in S, \quad (3.3)$$

$$f_{min}(\pi_s \cdot \langle s, t \rangle) = \min\{f_{min}(\pi_s), I(t)\}, \quad (3.4)$$

onde  $S$  é o próprio domínio da imagem.

Outra aplicação extensivamente modelada pela IFT é a Transformada de Watershed [7], muito empregada em tarefas de segmentação de imagens, a qual considera a imagem como sendo um mapa topográfico e simula a inundação desta superfície por fontes de água colocadas uma em cada mínimo regional; e uma barreira (linhas de watershed) sendo erguida toda vez que águas provenientes de fontes distintas se encontram, impedindo assim que eles se misturem. Utilizando um conjunto de sementes apropriado, isto é, sementes no objeto de interesse e no fundo da imagem, por exemplo, e a função de valor de caminho  $f_{max}$ , pode-se segmentar a imagem em objeto de interesse e fundo [27]. A Figura 3.3 ilustra este procedimento.

A maioria das aplicações envolvendo a IFT pode ser solucionada pela simples escolha de parâmetros seguida de um processamento local aplicado aos valores de custo de camin-

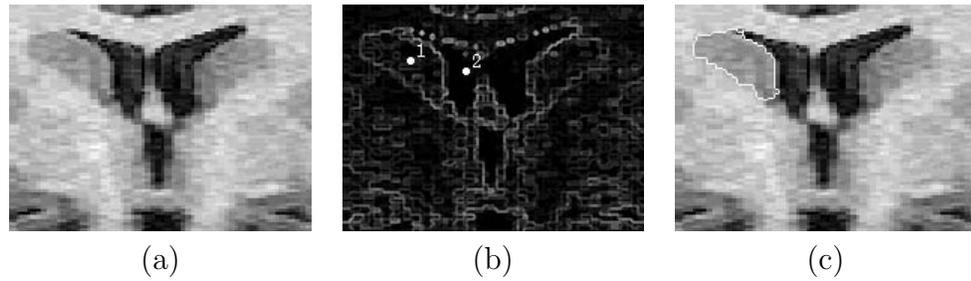


Figura 3.3: Exemplo de segmentação utilizando a IFT. (a) Uma imagem de ressonância magnética do cérebro. (b) Imagem de gradiente de (a) com uma semente selecionada dentro do núcleo caudado (1) e outra fora (2). (c) Imagem segmentada resultante.

hos, mapa de predecessores e mapa de raízes, em tempo proporcional ao número de pixels. Assim, a IFT unifica e estende várias técnicas de análise de imagens que, muito embora sejam baseadas em conceitos similares, são normalmente apresentadas como métodos não relacionados [27].

# Capítulo 4

## Classificadores baseados em floresta de Caminhos ótimos

Este capítulo tem por objetivo apresentar classificadores baseados em floresta de caminhos ótimos com aprendizado supervisionado e não-supervisionado. Nós modelamos o problema de reconhecimento de padrões como um problema de floresta de caminhos ótimos em um grafo definido no espaço de atributos, onde os nós são as amostras, as quais são representadas pelos seus respectivos vetores de atributos, e os arcos são definidos de acordo com uma relação de adjacência pré-estabelecida. Tanto os nós quanto os arcos podem ser ponderados, e diversas funções de custo podem ser empregadas com o intuito de particionar o grafo em árvores de caminhos ótimos, as quais são enraizadas pelos seus respectivos protótipos (sementes) na fase de treinamento. O rótulo de uma amostra a ser classificada é o mesmo do protótipo mais fortemente conexo a ela.

### 4.1 Classificação não-supervisionada

A presente seção tem por objetivo apresentar um método de classificação não-supervisionado baseado em floresta de caminhos ótimos, proposto inicialmente por Rocha et al. [63], o qual foi desenvolvido com o intuito de identificar clusters como sendo as árvores de uma floresta de caminhos ótimos. Embora não seja o intuito do presente trabalho de doutorado abordar classificadores não-supervisionados, uma apresentação do trabalho desenvolvido por Rocha et al. [63] faz-se necessária, pois alguns conceitos do mesmo foram utilizados para o desenvolvimento do classificador supervisionado baseado em OPF com grafo  $k$ -NN.

Nesta abordagem não-supervisionada, as amostras são modeladas como sendo os nós de um grafo, cujos arcos conectam os  $k$ -vizinhos mais próximos no espaço de atributos. O grafo é ponderado nos nós por valores de densidades originando, assim, uma função

de densidade de probabilidade (fdp), a qual é calculada levando-se em consideração as distâncias (peso dos arcos) entre os vetores de atributos de amostras adjacentes. O valor do melhor  $k$  é encontrado minimizando uma medida de corte em grafo e a maximização de uma função de valor de caminho origina uma floresta de caminhos ótimos, onde cada árvore (cluster) é enraizada em um máximo da fdp. As seções seguintes apresentam o método não-supervisionado baseado em floresta de caminhos ótimos (Seção 4.1.1) e a sua extensão para grandes bases de dados (Seção 4.1.2).

### 4.1.1 Fundamentação teórica

Seja  $Z$  uma base de dados tal que, para toda amostra  $s, t \in Z$ , existe um vetor de atributos  $\vec{v}(s)$ . Seja  $d(s, t)$  uma distância entre  $s$  e  $t$  no espaço de atributos. O problema fundamental na área de clustering é identificar grupos de amostras em  $Z$ , sendo que amostras de um mesmo grupo deveriam representar algum nível de semelhança de acordo com algum significado semântico.

Dizemos que uma amostra  $t$  é adjacente a uma amostra  $s$  (isto é,  $t \in A(s)$  ou  $(s, t) \in A$ ) quando alguma relação de adjacência é satisfeita. Por exemplo,

$$t \in A_1(s) \text{ se } d(s, t) \leq d_f \text{ ou} \quad (4.1)$$

$t \in A_2(s)$  se  $t$  é  $k$ -vizinho mais próximo de  $s$  no espaço de atributos, onde  $d_f$  e  $k > 1$  são parâmetros do tipo real e inteiro, respectivamente. Assim sendo, o par  $(Z, A_k)$  define então um grafo  $k$ -nn, onde  $A_k$  é uma relação de adjacência do tipo  $A_2$  e, posteriormente, do tipo  $A_3$  (Equação 4.3). Os arcos são ponderados por  $d(s, t)$  e os nós  $s \in Z$  são ponderados por um valor de densidade  $\rho(s)$ , dado por

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}(s)|} \sum_{t \in \mathcal{A}(s)} \exp\left(\frac{-d^2(s, t)}{2\sigma^2}\right), \quad (4.2)$$

onde  $\sigma = \frac{d_f}{3}$  e  $d_f$  é o comprimento do maior arco em  $(Z, A_k)$ . A escolha deste parâmetro considera todos os nós para o cálculo da densidade, assumindo que uma função gaussiana cobre a grande maioria das amostras com  $d(s, t) \in [0, 3\sigma]$ .

Relações de adjacência simétricas (Equação 4.1 por exemplo) resultam em relações de conectividade simétricas. Entretanto  $A_2$  na Equação ?? é uma relação de adjacência assimétrica. Dado que um máximo da fdp pode ser um subconjunto de amostras adjacentes com um mesmo valor de densidade, existe a necessidade da garantia da conectividade entre qualquer par de amostras naquele máximo. Assim, qualquer amostra deste conjunto de máximos pode ser representativa e alcançar outras amostras desse máximo e suas respectivas zonas de influência por um caminho ótimo. Isto requer uma modificação na

relação de adjacência  $A_2$ , para que a mesma seja simétrica nos platôs de  $\rho$  com o intuito de calcular os clusters:

$$\begin{aligned} \text{se } t &\in \mathcal{A}_2(s), \\ s &\notin \mathcal{A}_2(t) \text{ e} \\ \rho(s) &= \rho(t), \text{ então} \\ \mathcal{A}_3(t) &\leftarrow \mathcal{A}_2(t) \cup \{s\}. \end{aligned} \tag{4.3}$$

Se tivéssemos uma amostra por máximo, formando um conjunto  $S$  (pontos grandes na Figura 4.1a), então a maximização da função  $f_1$  resolveria o problema, ou seja:

$$\begin{aligned} f_1(\langle t \rangle) &= \begin{cases} \rho(t) & \text{se } t \in S \\ -\infty & \text{caso contrário} \end{cases} \\ f_1(\pi_s \cdot \langle s, t \rangle) &= \min\{f_1(\pi_s), \rho(t)\}. \end{aligned} \tag{4.4}$$

A função  $f_1$  possui um termo de inicialização e um termo de propagação, o qual associa a cada caminho  $\pi_t$  o menor valor de densidade ao longo do mesmo. Toda amostra  $t \in S$  define um caminho trivial  $\langle t \rangle$  devido ao fato de não ser possível alcançar  $t$  através de outro máximo da fdp sem passar através das amostras com valores de densidade menores que  $\rho(t)$  (Figura 4.1a). As amostras restantes iniciam com caminhos triviais de valor  $-\infty$  (Figura 4.1b), assim qualquer caminho oriundo de  $S$  possuirá valor maior. Considerando todos os caminhos possíveis de  $S$  a toda amostra  $s \notin S$ , o caminho ótimo  $P^*(s)$  será aquele cujo menor valor de densidade seja máximo.

Visto que não temos os máximos da fdp, a função de conectividade precisa ser escolhida de tal forma que seus valores iniciais  $h$  definam os máximos relevantes da fdp. Para  $f_1(\langle t \rangle) = h(t) < \rho(t)$ ,  $\forall t \in Z$ , alguns máximos da fdp serão preservados e outros serão alcançados por caminhos oriundos de outros máximos, cujos valores serão maiores do que seus valores iniciais. Por exemplo, se

$$\begin{aligned} h(t) &= \rho(t) - \delta, \\ \delta &= \min_{(s,t) \in \mathcal{A} | \rho(t) \neq \rho(s)} |\rho(t) - \rho(s)|, \end{aligned} \tag{4.5}$$

então todos os máximos de  $\rho$  serão preservados. Para altos valores de  $\delta$  os domos da fdp com altura menor que  $\delta$  não definirão zonas de influência.

Desejamos também evitar a divisão da zona de influência de um máximo em múltiplas zonas de influência, cada uma enraizada por uma amostra naquela máximo. Dado que o

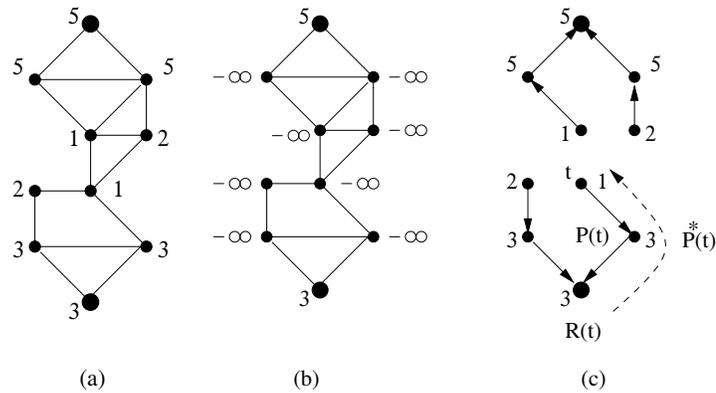


Figura 4.1: (a) Grafo cujos pesos dos nós são seus valores de fdp  $\rho(t)$ . Existem dois máximos com valores 3 e 5, respectivamente. Os pontos grandes indicam o conjunto de raízes  $S$ . (b) Valores de caminho triviais  $f_1(\langle t \rangle)$  para cada amostra  $t$ . (c) Floresta de caminhos ótimos  $P$  para  $f_1$  e os valores de caminho finais  $V(t)$ . O caminho ótimo  $P^*(t)$  (linha tracejada) pode ser obtido percorrendo os predecessores  $P(t)$  até a raiz  $R(t)$  para cada amostra  $t$ .

algoritmo da IFT primeiro identifica os máximos da fdp, antes de propagar suas zonas de influência, podemos modificá-lo de tal forma a detectar uma primeira amostra  $t$  para cada máximo, definindo o conjunto  $S$  em tempo real (*on-the-fly*). Nós então trocamos  $h(t)$  por  $\rho(t)$  e esta amostra irá conquistar as amostras restantes do mesmo máximo. Assim, a função de conectividade  $f_2$  final será dada por

$$f_2(\langle t \rangle) = \begin{cases} \rho(t) & \text{se } t \in S \\ h(t) & \text{caso contrário} \end{cases}$$

$$f_2(\pi_s \cdot \langle s, t \rangle) = \min\{f(\pi_s), \rho(t)\}. \quad (4.6)$$

O problema agora direciona-se em encontrar o melhor valor de  $k$  para definir  $A_k$ . A solução proposta por Rocha et al. [63] para encontrar o melhor  $k^*$  considera o corte mínimo no grafo provida pelos resultados do processo de clustering para  $k^* \in [1, k_{max}]$ , de acordo com a medida  $C(k)$  sugerida por Shi e Malik [64]:

$$C(k) = \sum_{i=1}^c \frac{W'_i}{W_i + W'_i}, \quad (4.7)$$

$$W_i = \sum_{\forall (s,t) \in \mathcal{A} | L(s)=L(t)=i} \frac{1}{d(s,t)}, \quad (4.8)$$

$$W'_i = \sum_{\forall (s,t) \in \mathcal{A} | L(s)=i, L(t) \neq i} \frac{1}{d(s,t)}, \quad (4.9)$$

onde  $L(t)$  é o rótulo da amostra  $t$ ,  $W'_i$  utiliza todos os pesos dos arcos entre o cluster  $i$  e os demais, e  $W_i$  utiliza todos os pesos dos arcos que pertencem ao cluster  $i = 1, 2, \dots, c$ . A Figura 4.2a mostra um exemplo com  $|Z| = 340$  amostras, as quais formam poucos clusters com diferentes concentrações de amostras no espaço de atributos 2D. Dependendo do valor de  $k$  escolhido, podemos encontrar até cinco agrupamentos de dados. Se  $k_{max} \geq 150$ , então o corte mínimo irá ocorrer quando todas as amostras estiverem agrupadas em um único cluster. O corte mínimo para  $k_{max} = 100$  identifica quatro clusters com o melhor  $k^* = 37$  (Figura 4.2b), e limitando a busca para  $k_{max} = 30$ , o corte mínimo identifica cinco clusters com melhor  $k^* = 29$  (Figura 4.2c).

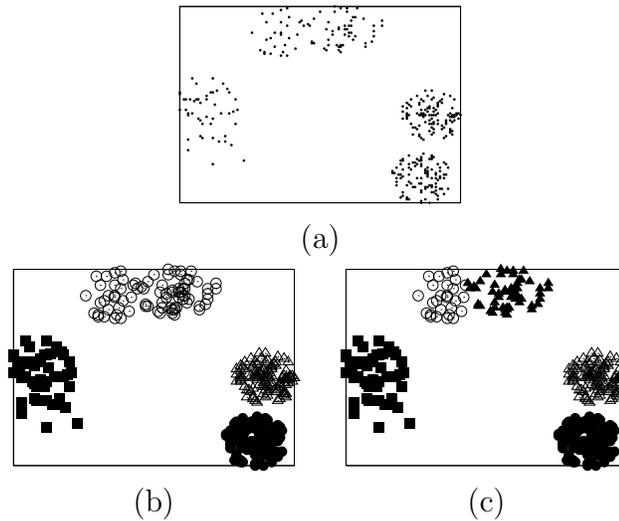


Figura 4.2: (a) Espaço de atributos com diferentes concentrações de amostras para cada cluster. Podemos identificar diferentes quantidades de clusters dependendo do valor de  $k$  escolhido. Soluções interessantes são (b) quatro e (c) cinco clusters.

Segue abaixo o algoritmo do classificador baseado em floresta de caminhos ótimos com aprendizado não-supervisionado.

### Algoritmo 2 – CLUSTERING POR FLORESTA DE CAMINHOS ÓTIMOS

ENTRADA: Grafo  $(Z, A_{k^*})$  e função  $\rho$ .

SAÍDA: Mapa de rótulos  $L$ , mapa de valores de caminho  $V$ , mapa de predecessores  $P$ .

AUXILIARES: Fila de prioridade  $Q$ , variáveis  $tmp$  e  $l \leftarrow 1$ .

1. **Para todo**  $s \in Z$ , **Faça**  $P(s) \leftarrow nil$ ,  $V(s) \leftarrow \rho(s) - \delta$ , *insira*  $s$  em  $Q$ .
2. **Enquanto**  $Q$  *é não vazia*, **Faça**
3.     *Remova de*  $Q$  *uma amostra*  $s$  *tal que*  $V(s)$  *é máximo*.
4.     **Se**  $P(s) = nil$ , **Então**
5.          $L(s) \leftarrow l$ ,  $l \leftarrow l + 1$ , e  $V(s) \leftarrow \rho(s)$ .

6. **Para cada**  $t \in A_{k^*}(s)$  e  $V(t) < V(s)$ , **Faça**
7.      $tmp \leftarrow \min\{V(s), \rho(t)\}$ .
8.     **Se**  $tmp > V(t)$ , **Então**
9.          $L(t) \leftarrow L(s)$ ,  $P(t) \leftarrow s$ ,  $V(t) \leftarrow tmp$ .
10.     **Atualize posição de**  $t$  em  $Q$ .

O Algoritmo 2 identifica uma raiz em cada máximo da fdp ( $P(s) = nil$  na Linha 4 implica que  $s \in S$ ), associa um rótulo distinto a cada raiz na Linha 5, e calcula a zona de influência (cluster) de cada raiz como sendo uma árvore de caminhos ótimos em  $P$ , tal que os nós de cada árvore recebem o mesmo rótulo que a sua raiz no mapa  $L$  (Linha 9). O algoritmo também retorna o mapa de valores de caminhos ótimos  $V$  e o mapa de predecessores  $P$ , sendo também mais robusto que o tradicional algoritmo de mean-shift [14], pois não depende de gradientes da fdp, utiliza um grafo  $k$ -nn e associa um rótulo para cada máximo, mesmo quando o máximo é composto por um componente conexo em  $(Z, A_{k^*})$ .

### 4.1.2 Extensão para grandes bases de dados

O Algoritmo 2 pode tornar-se proibitivo para grandes bases de dados, principalmente em aplicações que envolvem imagens 3D, pois a estimação do valor do melhor  $k$  requer o seu cálculo inúmeras vezes, aumentando ainda mais a complexidade do algoritmo. Capabianco et al. [11] propôs uma extensão do algoritmo de classificação não-supervisionado baseado em OPF para aplicações que possuem uma grande base de dados como, por exemplo, segmentação de substâncias branca e cinzenta do cérebro humano. Esta extensão é baseada em uma seleção aleatória de um conjunto  $Z' \subset Z$ . Seja  $V$  e  $L$  os mapas ótimos do Algoritmo 2 calculados no melhor grafo  $k$ -nn  $(Z', A_{k^*})$ . Uma amostra  $t \in Z \setminus Z'$  pode ser classificada como pertencente a um dos clusters simplesmente identificando qual raiz oferece o caminho ótimo como se esta amostra pertencesse à floresta original. Considerando os  $k$ -vizinhos mais próximos de  $t$  em  $Z'$ , podemos utilizar a Equação 4.2 para computar  $\rho(t)$ , avaliar os caminhos ótimos  $\pi_s \cdot \langle s, t \rangle$  e selecionar o que satisfaz

$$V(t) = \max_{\forall (s,t) \in A_{k^*}} \{\min\{V(s), \rho(t)\}\}. \quad (4.10)$$

Seja  $s^* \in Z'$  a amostra que satisfaz a Equação 4.10. O processo de classificação simplesmente associa  $L(s^*)$  como sendo o cluster de  $t$ .

## 4.2 Classificação supervisionada

Visto a constante necessidade de algoritmos de classificação de padrões aliada às deficiências existentes dos métodos apontados anteriormente, a presente seção apresenta duas novas metodologias para a tarefa de classificação supervisionada de padrões. As abordagens a serem apresentadas tratam as amostras como sendo os nós de um grafo, sendo os arcos definidos por uma relação de adjacência e ponderados por alguma métrica de distância aplicada a seus vetores de atributos, e diferem dos métodos tradicionais por não utilizar a idéia de geometria do espaço de atributos conseguindo, assim, melhores resultados em bases com outliers e sobreposição de classes. Duas abordagens supervisionadas foram estudadas, as quais diferem tanto na relação de adjacência e função de valor de caminho utilizadas, quanto na maneira de encontrar os protótipos: a primeira delas utiliza como relação de adjacência o grafo completo e busca como protótipos amostras que pertencem à intersecção entre as classes no conjunto de treinamento (Seção 4.2.1). A outra metodologia desenvolvida utiliza um grafo  $k$ -nn e encontra os protótipos como sendo os máximos regionais ou amostras de cada classe na junção entre as densidades de probabilidade (Seção 4.2.2). Vantagens e desvantagens de ambas as técnicas serão apresentadas nas seções seguintes.

### 4.2.1 Usando grafo completo

A técnica de classificação supervisionada baseada em florestas de caminhos ótimos apresentada nesta seção modela as amostras como sendo os nós de um grafo completo. Os elementos mais representativos de cada classe do conjunto de treinamento, isto é, os protótipos, são escolhidos como sendo os elementos pertencentes às regiões de fronteira entre as classes. Os protótipos participam de um processo de competição disputando as outras amostras oferecendo-lhes caminhos de menor custo e seus respectivos rótulos. Ao final deste processo, obtemos um conjunto de treinamento particionado em árvores de caminhos ótimos, sendo que a união das mesmas nos remete a uma floresta de caminhos ótimos. Esta abordagem apresenta vários benefícios com relação a outros métodos de classificação de padrões supervisionados: (i) é livre de parâmetros, (ii) possui tratamento nativo de problemas multiclassés e (iii) não faz alusão sobre forma e/ou separabilidade das classes.

O algoritmo OPF com grafo completo foi primeiramente apresentado por Papa et al. [56, 59] e tem sido amplamente utilizado em diversas aplicações, tais como avaliação de descritores de textura [50, 51], identificação de disfagias (dificuldade de deglutição) em seres humanos [65], diagnóstico automático de patologias na laringe [60] e classificação de impressões digitais [52]. Outra aplicação médica que utilizou esta técnica foi na área de parasitologia, onde o OPF com grafo completo foi aplicado na identificação de parasitos

intestinais em humanos, obtendo três solicitações de patentes: duas delas nacionais [29, 28] e a outra internacional [30].

As próximas seções irão discutir a fundamentação teórica e os algoritmos de treinamento e classificação do algoritmo baseado em OPF utilizando grafo completo.

### Fundamentação teórica

Seja  $Z$  uma base de dados  $\lambda$ -rotulada e  $Z_1$  e  $Z_3$  os conjuntos de treinamento e teste, respectivamente, com  $|Z_1|$  e  $|Z_3|$  amostras, as quais podem ser pixels/voxels/contornos tais que  $Z = Z_1 \cup Z_3$ . Seja  $\lambda(s)$  uma função que associa o rótulo correto  $i$ ,  $i = 1, 2, \dots, c$  da classe  $i$  a qualquer amostra  $s \in Z_1 \cup Z_3$ .

Seja  $S \in Z_1$  um conjunto de protótipos de todas as classes (isto é, amostras que melhor representam as classes). Seja  $\vec{v}$  um algoritmo que extrai  $n$  atributos (cor, forma e propriedades de textura) de qualquer amostra  $s \in Z_1 \cup Z_3$ , e retorna um vetor de atributos  $\vec{v}(s) \in \mathfrak{R}^n$ . A distância  $d(s, t)$  entre duas amostras,  $s$  e  $t$ , é dada pela distância entre seus vetores de atributos  $\vec{v}(s)$  e  $\vec{v}(t)$ .

Nosso problema consiste em usar  $S$ ,  $(\vec{v}, d)$  e  $Z_1$  para projetar um classificador ótimo, o qual pode prever o rótulo correto  $\lambda(s)$  de qualquer amostra  $s \in Z_3$ . Assim sendo, propomos um classificador que cria uma partição discreta ótima, a qual é uma floresta de caminhos ótimos computada em  $\mathfrak{R}^n$  pelo algoritmo da transformada imagem floresta [27].

Seja  $(Z_1, A)$  um grafo completo cujos nós são as amostras em  $Z_1$ , onde qualquer par de amostras define um arco em  $A$  (isto é,  $A = Z_1 \times Z_1$ ) (Figura 4.3a). Note que os arcos não precisam ser armazenados e o grafo não precisa ser explicitamente representado.

O algoritmo baseado em OPF pode ser utilizado com qualquer função de custo suave que pode agrupar amostras com propriedades similares [27]. Na versão OPF com grafo completo a função de custo abordada foi a  $f_{max}$  (Equação 3.1). O algoritmo baseado em OPF associa um caminho ótimo  $P^*(s)$  de  $S$  a toda amostra  $s \in Z_1$ , formando uma floresta de caminhos ótimos  $P$  (uma função sem ciclos, a qual associa a todo  $s \in Z_1$  seu predecessor  $P(s)$  em  $P^*(s)$ , ou uma marca *nil* quando  $s \in S$ , como mostrado na Figura 4.3d). Seja  $R(s) \in S$  a raiz de  $P^*(s)$  a qual pode ser alcançada por  $P(s)$ . O algoritmo computa, para cada  $s \in Z_1$ , o custo  $V(s)$  de  $P^*(s)$ , o rótulo  $L(s) = \lambda(R(s))$  e o seu predecessor  $P(s)$ , como segue.

#### **Algoritmo 3** – CLASSIFICADOR SUPERVISIONADO BASEADO EM FLORESTA DE CAMINHOS ÓTIMOS USANDO GRAFO COMPLETO

- ENTRADA: Um conjunto de treinamento  $Z_1$   $\lambda$ -rotulado, protótipos  $S \subset Z_1$  e o par  $(v, d)$  para vetor de atributos e cálculo das distâncias.
- SAÍDA: Floresta de caminhos ótimos  $P$ , mapa de valores de custo de caminhos  $V$  e mapa de rótulos  $L$
- AUXILIARES: Fila de prioridades  $Q$ , e variável  $cost$ .

1. **Para todo**  $s \in Z_1$ , **Faça**  $P(s) \leftarrow nil$  e  $V(s) \leftarrow +\infty$ .
2. **Para todo**  $s \in S$ , **Faça**  $V(s) \leftarrow 0$ ,  $P(s) \leftarrow nil$ ,  $L(s) = \lambda(s)$  e *insira*  $s$  em  $Q$ .
3. **Enquanto**  $Q$  *é não vazia*, **Faça**
  4. *Remova de*  $Q$  *uma amostra*  $s$  *tal que*  $V(s)$  *é mínimo*.
  5. **Para cada**  $t \in Z_1$  *tal que*  $s \neq t$  e  $V(t) > V(s)$ , **Faça**
    6. *Calcule*  $cst \leftarrow \max\{V(s), d(s, t)\}$ .
    7. **Se**  $cst < V(t)$ , **Então**
      8. **Se**  $V(t) \neq +\infty$ , **Então** *remova*  $t$  *de*  $Q$ .
      9.  $P(t) \leftarrow s$ ,  $L(t) \leftarrow L(s)$  e  $V(t) \leftarrow cst$ .
      10. *Insira*  $t$  *em*  $Q$ .

As linhas 1–2 inicializam os mapas e inserem protótipos em  $Q$ . O laço principal calcula um caminho ótimo de  $S$  para cada amostra  $s \in Z_1$  em uma ordem não decrescente de custos (linhas 3–10). A cada iteração um caminho de custo de ótimo  $V(s)$  é obtido em  $P$  (linha 4). Empates são resolvidos em  $Q$  utilizando a política FIFO (first-in-first-out), ou seja, quando dois caminhos atingem uma determinada amostra  $s$  com o mesmo custo mínimo,  $s$  é associado ao primeiro caminho que o atingiu. O restante das linhas avalia se o caminho que atinge uma amostra adjacente  $t$  através de  $s$  é mais barato que o caminho que termina em  $t$ . Caso positivo, atualiza  $Q$ ,  $P(t)$ ,  $L(t)$  e  $V(t)$ . No final do algoritmo,  $V$  armazena o valor do custo do caminho ótimo de  $S$  a cada amostra  $s \in Z_1$  de acordo com  $f_{max}$ .

O algoritmo OPF com grafo completo utilizando  $f_{max}$  pode ser entendido como uma transformada de Watershed usando a IFT [48] computada no espaço de atributos  $n$ -dimensional (nota-se a semelhança do algoritmo OPF com o algoritmo da IFT—Algoritmo 1). Além dessa extensão, outras contribuições significativas são os processos de treinamento e aprendizado, o qual encontra protótipos ótimos (marcadores) na fronteira entre as classes afastando *outliers* (amostras de uma determinada classe que estão presentes em uma região de outra classe no espaço de atributos) do conjunto de treinamento, aumentando a acurácia do classificador.

## Treinamento

A fase de treinamento do classificador baseado em floresta de caminhos ótimos usando o grafo completo consiste, basicamente, em encontrar o conjunto  $S$  de protótipos, ou seja, os elementos mais representativos de cada classe. Várias heurísticas poderiam ser adotadas como, por exemplo, uma escolha aleatória de protótipos. Entretanto, tal escolha pode prejudicar o desempenho do classificador, tornando-o instável e com um alto grau de sensibilidade com relação aos protótipos escolhidos. Desejamos, assim, estimar protótipos

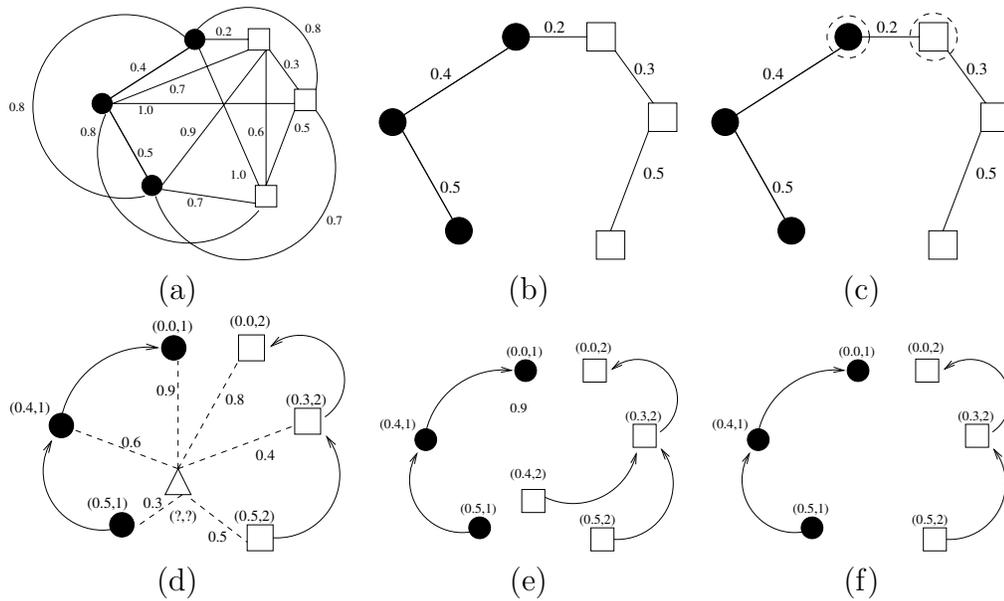


Figura 4.3: (a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) MST do grafo completo. (c) Protótipos escolhidos como sendo os elementos adjacentes de classes diferentes na MST (nós circulados). (d) Floresta de caminhos ótimos resultante para a função de valor de caminho  $f_{max}$  e dois protótipos. Os identificadores  $(x, y)$  acima dos nós são, respectivamente, o custo e o rótulo dos mesmos. A seta indica o nó predecessor no caminho ótimo. (e) Uma amostra de teste (triângulo) da classe 2 e suas conexões (linhas pontilhadas) com os nós do conjunto de treinamento. (f) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2 e o custo de classificação 0.4 são associados à amostra de teste. Note que, mesmo a amostra de teste estando mais próxima de um nó da classe 1, ela foi classificada como sendo da classe 2.

nas regiões de sobreposição de amostras e nas fronteiras entre as classes, visto que são regiões muito susceptíveis a erros de classificação.

Computando uma MST no grafo completo  $(Z_1, A)$ , obtemos um grafo conexo acíclico cujos nós são todas as amostras em  $Z_1$ , e os arcos são não direcionados e ponderados (Figura 4.3b). Seus pesos são dados pela distância  $d$  entre os vetores de atributos de amostras adjacentes. Esta árvore de espalhamento é ótima no sentido em que a soma dos pesos de seus arcos é mínima se comparada a outras árvores de espalhamento no grafo completo. Os protótipos a serem escolhidos são os elementos conectados na MST com diferentes rótulos em  $Z_1$ , isto é, elementos mais próximos de classes diferentes (Figura 4.3c). Removendo-se os arcos entre classes diferentes, tais amostras adjacentes tornam-se protótipos em  $S$  e o Algoritmo 3 pode computar uma floresta de caminhos ótimos em  $Z_1$  (Figura 4.3d). Note que uma dada classe pode ser representada por múltiplos protótipos (isto é, árvores de caminhos ótimos) e deve existir pelo menos

um protótipo por classe.

O Algoritmo 3 pode computar uma floresta de caminhos ótimos com erro zero de classificação em  $Z_1$ , desde que a função  $f_{max}$  seja modificada. A idéia consiste, basicamente, em ponderar os arcos entre amostras de diferentes classes com um valor muito alto, impossibilitando assim que protótipos de uma classe conquistem elementos de outras classes (similar procedimento também é utilizado pelo classificador baseado em OPF usando grafo  $k$ -nn, descrito na Seção 4.2.2). Desta forma, a função de valor de caminho  $f_{max}$  poderia ser escrita da seguinte forma:

$$\begin{aligned} f_{max}(\langle t \rangle) &= \begin{cases} 0 & \text{se } t \in \mathcal{S} \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi_s \cdot \langle s, t \rangle) &= \begin{cases} +\infty & \text{se } \lambda(t) \neq \lambda(s) \\ \max\{f_{max}(\pi_s), d(s, t)\} & \text{caso contrário.} \end{cases} \end{aligned} \quad (4.11)$$

A Equação 4.11 pondera todos os arcos  $(s, t) \in A_k$  tais que  $\lambda(t) \neq \lambda(s)$  com  $d(s, t) = +\infty$ , impedindo que tais arcos pertençam a algum caminho ótimo. Entretanto, vale destacar que, embora a Equação 4.11 garanta erro zero de classificação no conjunto de treinamento, a mesma produz resultados similares aos obtidos usando a Equação 3.1.

### Classificação

Para qualquer amostra  $t \in Z_3$ , consideramos todos os arcos conectando  $t$  com amostras  $s \in Z_1$ , tornando  $t$  como se fosse parte do grafo original (ver Figura 4.3e, onde a amostra  $t$  é representada pelo triângulo no grafo). Considerando todos os possíveis caminhos entre  $S$  e  $t$ , desejamos encontrar o caminho ótimo  $P^*(t)$  de  $S$  até  $t$  com a classe  $\lambda(R(t))$  de seu protótipo  $R(t) \in S$  mais fortemente conexo. Este caminho pode ser identificado incrementalmente, avaliando o valor do custo ótimo  $V(t)$  como

$$V(t) = \min\{\max\{V(s), d(s, t)\}\}, \forall s \in Z_1. \quad (4.12)$$

Seja  $s^* \in Z_1$  o nó que satisfaz a equação acima (isto é, o predecessor  $P(t)$  no caminho ótimo  $P^*(t)$ ). Dado que  $L(s^*) = \lambda(R(t))$ , a classificação simplesmente associa  $L(s^*)$  como a classe de  $t$  (Figura 4.3f). Um erro ocorre quando  $L(s^*) \neq \lambda(t)$ . Um procedimento similar é aplicado para amostras no conjunto de avaliação  $Z_2$  (Seção 4.2.3). Neste caso, entretanto, gostaríamos de usar as amostras em  $Z_2$  com o intuito de aprender a distribuição das classes no espaço de atributos e, conseqüentemente, melhorar o desempenho de classificadores baseados em OPF.

Vale ressaltar que, embora a amostra a ser classificada esteja mais próxima de um elemento da classe bola (Figura 4.3e), a mesma é classificada como sendo da classe quadrado,

o que demonstra que os classificadores baseados em OPF utilizam a força de conectividade entre as amostras para a classificação dos dados, ou seja, não são algoritmos baseados em conectividade local apenas como, por exemplo, os classificadores NN e  $k$ -NN.

Ressaltamos também que o classificador baseado em OPF com grafo completo assemelha-se ao algoritmo NN somente no caso onde todos os elementos do conjunto de treinamento são considerados protótipos, sendo este um caso atípico onde, certamente, o conjunto de atributos escolhido não foi o mais adequado para a representação do conjunto de dados. Outra diferença a ser considerada é que os classificadores NN e  $k$ -NN tomam uma decisão local para a classificação dos dados, ao contrário dos classificadores baseados em OPF, os quais possibilitam uma solução em âmbito global, criando uma floresta de caminhos ótimos que mapeia, para cada amostra do conjunto de dados, o caminho ótimo entre ela e o seu protótipo mais fortemente conexo

### 4.2.2 Usando grafo $k$ -nn

A técnica de classificação supervisionada baseada em floresta de caminhos ótimos apresentada nesta seção modela as amostras novamente como sendo os nós de um grafo [58]. Entretanto, a relação de adjacência utilizada é a dos  $k$ -vizinhos mais próximos e tanto os arcos quanto os nós são ponderados. A diferença básica entre esta abordagem e a previamente apresentada (Seção 4.2.1), é o fato de a abordagem que faz uso do grafo completo estimar protótipos na fronteira das classes, enquanto a metodologia a ser apresentada aqui estima os protótipos nos pontos de alta concentração de amostras.

A principal motivação desta representação seria o estudo de outras metodologias para classificadores supervisionados baseados em florestas de caminhos ótimos, utilizando outras funções de custo, relações de adjacência e maneiras diferentes de se encontrar os protótipos. As definições, bem como os algoritmos desta abordagem, são apresentados nas próximas seções.

#### Fundamentação teórica

Seja  $Z$  uma base de dados  $\lambda$ -rotulada e  $Z_1$  e  $Z_3$  os conjuntos de treinamento e teste, respectivamente, com  $|Z_1|$  e  $|Z_3|$  amostras, tais que  $Z = Z_1 \cup Z_3$ . Dado um grafo  $(Z_1, A_k)$ , o algoritmo OPF com grafo  $k$ -nn particiona o conjunto  $Z_1$  em uma floresta de caminhos ótimos de acordo com uma função de densidade de probabilidade (fdp).

Nesta abordagem temos um grafo ponderado nos vértices e nos arcos. A distância  $d(s, t)$  entre duas amostras  $s, t \in Z_1$  define o peso do arco  $(s, t)$ . Os nós são ponderados pelos seus valores de densidade de probabilidade, os quais são computados usando o peso dos arcos (Equação 4.2). A função de valor de caminho dada pela Equação 4.6 associa a um nó terminal  $s$  de cada caminho o mínimo entre os valores de densidade ao longo do mesmo

e um valor de densidade inicial. O caminho de valor máximo para cada amostra  $s$ , a partir de um conjunto de elementos protótipos (raízes), particiona o conjunto de treinamento  $Z_1$  em uma floresta de caminhos ótimos. As raízes da floresta formam um subconjunto dos máximos da fdp onde, cada raiz, define uma árvore de caminhos ótimos (zona de influência do respectivo máximo) composta pelas suas amostras mais fortemente conexas, as quais recebem o mesmo rótulo de sua raiz, como pode ser demonstrado pelo Algoritmo 4. Esta abordagem é muito semelhante ao algoritmo da versão não-supervisionada do OPF (Algoritmo 2). A principal diferença é que agora o conjunto de entrada é  $\lambda$ -rotulado.

**Algoritmo 4** – CLASSIFICADOR SUPERVISIONADO BASEADO EM FLORESTA DE CAMINHOS ÓTIMOS USANDO GRAFO  $k$ -NN

ENTRADA: Grafo  $k$ -nn  $(Z_1, A_k)$ ,  $\lambda(s)$  para todo  $s \in Z_1$  e função de valor de caminho  $f_1$ .

SAÍDA: Mapa de rótulos  $L$ , mapa de valores de caminho  $V$  e a floresta de caminhos ótimos  $P$ .

AUXILIARES: Fila de prioridade  $Q$  e a variável  $tmp$ .

1. **Para todo**  $s \in Z_1$ , **Faça**  $P(s) \leftarrow nil$ ,  $V(s) \leftarrow \rho(s) - \delta$ ,  $L(s) \leftarrow \lambda(s)$  e *insira*  $s$  em  $Q$ .
2. **Enquanto**  $Q$  é não vazia, **Faça**
3.     Remova de  $Q$  uma amostra  $s$  tal que  $V(s)$  é máximo.
4.     **Se**  $P(s) = nil$ , **Então**
5.         **L**  $V(s) \leftarrow \rho(s)$ .
6.     **Para cada**  $t \in A_k(s)$  e  $V(t) < V(s)$ , **Faça**
7.         **L**  $tmp \leftarrow \min\{V(s), \rho(t)\}$ .
8.         **Se**  $tmp > V(t)$ , **Então**
9.             **L**  $L(t) \leftarrow L(s)$ ,  $P(t) \leftarrow s$ ,  $V(t) \leftarrow tmp$ .
10.         **L** *Atualize posição de*  $t$  em  $Q$ .

Inicialmente (Linha 1), todos os caminhos são triviais com valores  $f_1(\langle t \rangle) = \rho(t) - \delta$ . Os máximos globais da fdp são os primeiros elementos a serem removidos de  $Q$ , os quais são identificados como sendo as raízes da floresta pelo teste  $P(s) = nil$  na linha 4, onde atualizamos o seu valor de caminho correto  $f_1(\langle t \rangle) = V(s) = \rho(t)$ . Cada nó  $s$  removido de  $Q$  oferece um caminho  $\pi_s \cdot \langle s, t \rangle$  para cada nó  $t$  adjacente no laço da Linha 6 até a Linha 10. Se o valor de caminho  $f_1(\pi_s \cdot \langle s, t \rangle) = \min\{V(s), \rho(t)\}$  (Linha 7) é melhor que o valor de caminho atual  $f_1(\pi_t) = V(t)$  (Linha 8), então  $\pi_t$  é atualizado para  $\pi_s \cdot \langle s, t \rangle$  (isto é,  $P(s) \leftarrow s$ , e o valor do caminho e o rótulo de  $t$  são atualizados (Linha 9). Máximos locais da fdp são também identificados como sendo raízes durante a execução do algoritmo, o qual tem como saída o mapa de rótulos  $L$ , mapa de valores de caminho  $V$  e a floresta de caminhos ótimos  $P$ .

### Treinamento

Um erro de classificação no conjunto de treinamento ocorre quando  $L(t) \neq \lambda(t)$ . Assim, definimos o melhor valor de  $k^* \in [1, k_{max}]$  como sendo aquele que maximiza a acurácia da classificação no conjunto de treinamento, dada pela Equação 4.16.

É esperado que cada classe seja representada por pelo menos um máximo da fdp e  $L(t) = \lambda(t)$  para todo  $t \in Z_1$  (erro zero de classificação no conjunto de treinamento). Entretanto, essas propriedades não podem ser garantidas com a função de valor de caminho  $f_2$  e o melhor valor  $k^*$ . Com o intuito de assegurar tais propriedades, primeiro encontramos  $k^*$  utilizando  $f_2$  e então executamos o Algoritmo 4 mais uma vez usando a função de valor de caminho  $f_3$  ao contrário de  $f_2$ , dada por

$$f_3(\langle t \rangle) = \begin{cases} \rho(t) & \text{se } t \in \mathcal{S} \\ \rho(t) - \delta & \text{caso contrário} \end{cases}$$

$$f_3(\pi_s \cdot \langle s, t \rangle) = \begin{cases} -\infty & \text{se } \lambda(t) \neq \lambda(s) \\ \min\{f_2(\pi_s), \rho(t)\} & \text{caso contrário.} \end{cases} \quad (4.13)$$

A Equação 4.13 pondera todos os arcos  $(s, t) \in A_k$  tais que  $\lambda(t) \neq \lambda(s)$  com  $d(s, t) = -\infty$ , impedindo que tais arcos pertençam a algum caminho ótimo. O processo de treinamento acima é descrito pelo Algoritmo 5.

### Algoritmo 5 – TREINAMENTO

- ENTRADA: Conjunto de treinamento  $Z_1$ ,  $\lambda(s)$  para todo  $s \in Z_1$ ,  $k_{max}$  e funções de valor de caminho  $f_2$  e  $f_3$ .
- SAÍDA: Mapa de rótulos  $L$ , mapa de valores de caminho  $V$  e a floresta de caminhos ótimos  $P$ .
- AUXILIARES: Variáveis  $i$ ,  $k$ ,  $k^*$ ,  $MaxAcc \leftarrow -\infty$ ,  $Acc$  e vetores  $FP$  e  $FN$  de tamanho  $c$ .

1. Para  $k = 1$  até  $k_{max}$  faça
2.     Crie um grafo  $(Z_1, A_k)$  ponderado nos nós através da Equação 4.2.
3.     Calcule  $(L, V, P)$  utilizando o Algoritmo 4 com  $f_2$ .
4.     Para cada classe  $i = 1, 2, \dots, c$ , faça
5.          $FP(i) \leftarrow 0$  e  $FN(i) \leftarrow 0$ .
6.     Para cada amostra  $t \in Z_1$ , faça
7.         Se  $L(t) \neq \lambda(t)$ , então
8.              $FP(L(t)) \leftarrow FP(L(t)) + 1$ .
9.              $FN(\lambda(t)) \leftarrow FN(\lambda(t)) + 1$ .
10.     Calcule  $Acc$  através da Equação 4.16.
11.     Se  $Acc > MaxAcc$ , então
12.          $k^* \leftarrow k$  e  $MaxAcc \leftarrow Acc$ .

13.  $\perp$  Destrua o grafo  $(Z_1, A_k)$ .
14. Crie o grafo  $(Z_1, A_{k^*})$  ponderado nos nós através da Equação 4.2.
15. Calcule  $(L, V, P)$  utilizando o Algoritmo 4 com  $f_3$ .

### Classificação

Uma amostra  $t \in Z_3$  pode ser associada a uma dada classe  $i, i = 1, 2, \dots, c$  simplesmente identificando qual raiz (protótipo) oferece o caminho ótimo como se esta amostra pertencesse à floresta. Considerando os  $k$ -vizinhos mais próximos de  $t$  em  $Z_3$ , podemos utilizar a Equação 4.2 para computar  $\rho(t)$ , avaliar os caminhos ótimos  $\pi_s \cdot \langle s, t \rangle$  e selecionar o que satisfaz a Equação 4.10. O rótulo de  $t$  será o mesmo rótulo do seu predecessor  $P(t)$ , ou seja,  $L(t) = L(P(s))$ . Um erro ocorre quando  $L(t) \neq \lambda(t)$ . A Figura 4.4 ilustra esse processo.

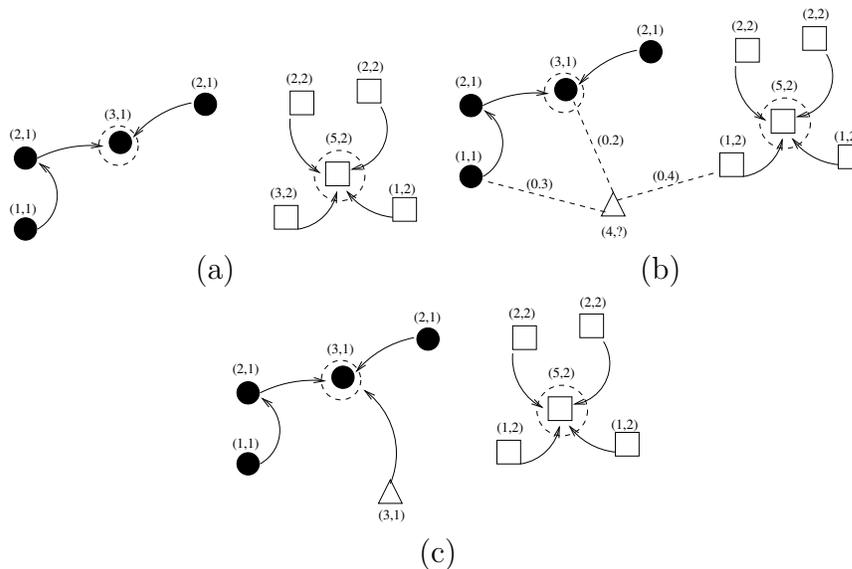


Figura 4.4: (a) Floresta de caminhos ótimos obtida através do Algoritmo 4, cujos elementos são ponderados nos nós com seus respectivos valores de densidade. Os indicadores  $(x, y)$  acima dos nós são, respectivamente, o seu valor de densidade e o rótulo da classe a qual ele pertence. Os elementos circulados representam os máximos de cada classe. (b) Processo de classificação, onde um nó  $t \in Z_3$  (triângulo) a ser classificado é inserido no grafo e conectado aos seus  $k$ -vizinhos mais próximos ( $k = 3$  no exemplo), e o seu valor de densidade é calculado. (c) Processo final da classificação, no qual a amostra  $t$  é classificada com o rótulo da amostra  $s \in Z_1$  que satisfaz a Equação 4.10, ou seja,  $L(t) = L(s)$ . No exemplo acima, o elemento a ser classificado foi conquistado pelo máximo da classe 1.

A Figura 4.4a ilustra uma floresta de caminhos ótimos obtida através do Algoritmo 4,

cujos elementos são ponderados nos nós com seus respectivos valores de densidade. A Figura 4.4b apresenta o processo de classificação, onde um nó  $t \in Z_3$  (triângulo) a ser classificado é inserido no grafo e conectado aos seus  $k$ -vizinhos mais próximos ( $k = 3$  no exemplo), e o seu valor de densidade é calculado (seu rótulo é ainda desconhecido). A Figura 4.4c ilustra o processo final da classificação, no qual a amostra  $t$  é classificada com o rótulo da amostra  $s \in Z_1$  que satisfaz a Equação 4.10, ou seja,  $L(t) = L(s)$ . No exemplo acima, o elemento a ser classificado foi conquistado pelo máximo da classe 1.

Embora o classificador baseado em OPF usando grafo  $k$ -nn possa parecer semelhante a um classificador Bayesiano (BC) com fdp gaussiana, algumas diferenças podem ser elencadas: (i) a fase de treinamento do BC simplesmente consiste em, para cada amostra, calcular a sua densidade com base nos  $k$  vizinhos mais próximos ao passo que, no classificador baseado em OPF, as amostras com maior valor de densidade viram as raízes para o cálculo da floresta de caminhos ótimos, (ii) a fase de teste do BC consiste em, dada uma amostra  $t$ , calcular  $P(t|w_i)$ ,  $i = 1, 2, \dots, c$  e decidir pela classe  $w_i$  cujo  $P(w_i|t) > P(w_j|t)$ ,  $i \neq j$  (Regra de Bayes), onde  $P(w_j|t) = \frac{P(t|w_j)P(w_j)}{P(t)}$ . Ocorre que, em bases de dados com muitas classes, o cálculo de  $P(t|w_i)$ ,  $i = 1, 2, \dots, c$ , gera um custo computacional alto, ao passo que no classificador baseado em OPF com grafo  $k$ -nn a fase de teste consiste, essencialmente, em conectar a amostra  $t$  aos  $k$ -vizinhos mais próximos, calcular a densidade  $P(t)$  e verificar qual protótipo oferece o caminho ótimo. As Figuras 4.5a e 4.5b ilustram, respectivamente, o processo de classificação de uma amostra  $t$  para os classificadores baseados em OPF com grafo  $k$ -nn e BC.

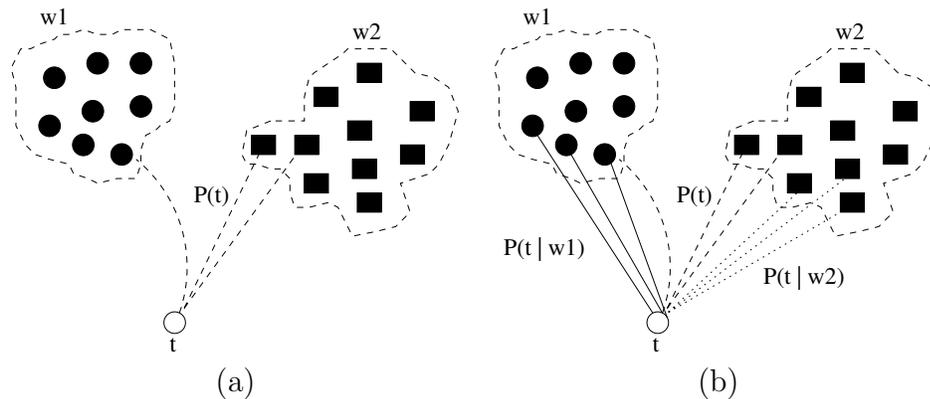


Figura 4.5: Processo de classificação de uma amostra  $t$ . (a) Classificador baseado em OPF com grafo  $k$ -nn conecta a amostra  $t$  aos seus  $k$ -vizinhos para obter  $P(t)$ . (b) BC conecta  $t$  aos  $k$ -vizinhos da classe  $w_1$  (linha sólida) para obter  $P(t|w_1)$  e depois conecta  $t$  aos  $k$ -vizinhos da classe  $w_2$  (linha pontilhada) para obter  $P(t|w_2)$ . Em seguida, conecta  $t$  aos  $k$ -vizinhos mais próximos (linha tracejada) para obter  $P(t)$  e, conseqüentemente, usar a Regra de Bayes para classificar a amostra  $t$ .

### 4.2.3 Aprendizado

Seja  $Z$  uma base de dados e  $Z_1$ ,  $Z_2$  e  $Z_3$  os conjuntos de treinamento, avaliação e teste, respectivamente, com  $|Z_1|$ ,  $|Z_2|$  e  $|Z_3|$  amostras. Esta seção mostra como utilizar um terceiro conjunto de avaliação  $Z_2$  para melhorar a composição das amostras em  $Z_1$  sem aumentar o seu tamanho. Apresentamos também um algoritmo de treinamento genérico, o qual troca amostras classificadas incorretamente em  $Z_2$  por amostras selecionadas aleatoriamente de  $Z_1$  (com algumas restrições) durante algumas iterações) [59]. Este procedimento assume que as amostras mais informativas podem ser obtidas através dos erros. Os valores de acurácia ao longo das iterações definem uma *curva de aprendizagem*. Caso as amostras classificadas incorretamente forem informativas, então a acurácia do classificador, re-treinado com o próximo conjunto  $Z_1$ , deve aumentar no conjunto  $Z_2$ . Ao final do processo, selecionamos a instância do classificador que obteve a maior exatidão para testá-lo em  $Z_3$ . Comparando as acurácias dos classificadores em  $Z_3$ , antes e depois do processo de treinamento, podemos avaliar suas capacidades de aprendizagem com os erros. Este procedimento foi primeiramente apresentado por Papa et al. [59].

A acurácia  $Acc$  é calculada levando em consideração que as classes podem ter diferentes tamanhos em  $Z_3$  (definição similar à aplicada para  $Z_2$ ). Caso tivéssemos duas classes, por exemplo, com diferentes tamanhos e um classificador sempre associasse o rótulo da classe com mais representantes, sua acurácia diminuiria devido a alta taxa de erro na classe com menor número de elementos.

Seja  $NZ_3(i)$ ,  $i = 1, 2, \dots, c$ , o número de elementos em  $Z_3$  de cada classe  $i$ . Definimos

$$e_{i,1} = \frac{FP(i)}{|Z_3| - |NZ_3(i)|} \quad \text{and} \quad e_{i,2} = \frac{FN(i)}{|NZ_3(i)|}, \quad i = 1, \dots, c \quad (4.14)$$

onde  $FP(i)$  e  $FN(i)$  são os falsos positivos e falsos negativos, respectivamente. Isto significa que  $FP(i)$  corresponde ao número de amostras de outras classes que foram classificadas como sendo da classe  $i$  em  $Z_3$ , e  $FN(i)$  é o número de amostras da classe  $i$  que foram incorretamente classificadas como sendo de outras classes em  $Z_3$ . Os erros  $e_{i,1}$  e  $e_{i,2}$  são usados para definir

$$E(i) = e_{i,1} + e_{i,2}, \quad (4.15)$$

onde  $E(i)$  é a soma parcial dos erros da classe  $i$ . Finalmente, a acurácia  $Acc$  de cada classificação é dada por

$$Acc = \frac{2c - \sum_{i=1}^c E(i)}{2c} = 1 - \frac{\sum_{i=1}^c E(i)}{2c}. \quad (4.16)$$

O Algoritmo 6 retorna uma curva de aprendizagem e a instância do classificador com a maior acurácia em  $Z_2$  após  $T$  iterações, podendo ser utilizado para diferentes classificadores. Neste trabalho, o Algoritmo 6 foi empregado para os classificadores baseados em OPF, SVM, ANN-MLP,  $k$ -NN e BC, apenas trocando as Linhas 3 e 16 – 17.

**Algoritmo 6** – APRENDIZADO

- ENTRADA: Conjuntos de treinamento e avaliação rotulados por  $\lambda$ ,  $Z_1$  e  $Z_2$ , número  $T$  de iterações, e o par  $(v, d)$  para vetor de atributos e cálculo das distâncias.
- SAÍDA: Curva de aprendizagem  $\mathcal{L}$  e o classificador baseado em OPF que obteve a maior acurácia sobre  $Z_2$ .
- AUXILIARES: Vetores de falsos positivos e falsos negativos,  $FP$  e  $FN$ , de tamanho  $c$ , lista  $LM$  de amostras classificadas incorretamente e variável  $Acc$ .
1. **Para cada** iteração  $I = 1, 2, \dots, T$ , **Faça**
  2.      $LM \leftarrow \emptyset$
  3.     Treine OPF/SVM/ANN-MLP/ $k$ -NN/BC com  $Z_1$ .
  4.     **Para cada** classe  $i = 1, 2, \dots, c$ , **Faça**
  5.          $FP(i) \leftarrow 0$  e  $FN(i) \leftarrow 0$ .
  6.     **Para cada** amostra  $t \in Z_2$ , **Faça**
  7.         Use o classificador obtido na Linha 3 para classificar  $t$  com o rótulo  $L(t)$ .
  8.         **Se**  $L(t) \neq \lambda(t)$ , **Então**
  9.              $FP(L(t)) \leftarrow FP(L(t)) + 1$ .
  10.             $FN(\lambda(t)) \leftarrow FN(\lambda(t)) + 1$ .
  11.             $LM \leftarrow LM \cup t$ .
  12.     Calcule  $Acc$  pela Equação 4.16 e salve a instância atual do classificador.
  13.      $\mathcal{L}(I) \leftarrow Acc$
  14.     **Enquanto**  $LM \neq \emptyset$ , **Faça**
  15.          $LM \leftarrow LM \setminus t$
  16.         Troque  $t$  por um objeto da mesma classe selecionado aleatoriamente
  17.         em  $Z_1$ , sob certas restrições.
  18. Seleccione a instância do classificador com a maior acurácia  $\mathcal{L}(\mathcal{I})$ .

Para o OPF, a Linha 3 é implementada calculando  $S^* \subset Z_1$  como descrito na Seção 4.2.1, sendo que os mapas de predecessores  $P$ , rótulos  $L$  e valores de custo de caminho  $V$  são encontrados executando o Algoritmo 3. A classificação é feita encontrando  $s^* \in Z_1$  que satisfaz a Equação 4.12, isto é,  $P(t) = s^*$  e  $L(t) \leftarrow L(s^*)$ . As restrições nas Linhas 16–17 referem-se a manter os protótipos fora do processo de troca entre os conjuntos  $Z_1$  e  $Z_2$ . Fazemos o mesmo com os vetores de suporte no caso da SVM. Contudo, eles podem ser selecionados em iterações futuras caso os mesmos não sejam mais protótipos ou vetores de suporte. Para o algoritmo SVM, utilizamos o pacote de implementação LibSVM [13] com kernel de função de base radial, otimização de parâmetros e estratégia OVO para solucionar o problema de várias classes para implementar a linha 3.

Utilizamos a biblioteca FANN (*Fast Artificial Neural Network Library*) [54] para implementar a ANN-MLP. A configuração da rede é dada por  $x:y:z$ , onde  $x = n$  (número de

atributos),  $y = |Z_1| - 1$  e  $z = c$  (número de classes) correspondem ao número de neurônios nas camadas de entrada, escondida e de saída, respectivamente [38]. Na linha 3, a ANN-MLP é treinada pelo algoritmo de retro-propagação. Não é empregada nenhuma restrição nas Linhas 16 – 17. Entretanto, mantemos os pesos dos neurônios como configuração inicial para o treinamento da próxima iteração. Para o algoritmos  $k$ -NN e BC, o treinamento da linha 3 envolve o cálculo de valor de  $k$  que obtém a maior acurácia em  $Z_2$ . Já para a versão do algoritmo de  $k$ -NN que não utiliza aprendizagem, obtemos o valor de  $k$  que maximiza a acurácia em  $Z_1$  de acordo com a abordagem Leave-One-Out [44]. As Linhas 16 – 17 são implementadas sem restrições para ambos classificadores  $k$ -NN e BC. Para a versão do algoritmo BC que não utiliza aprendizagem, obtemos o valor de  $k$  que maximiza a acurácia em  $Z_1$ .

As Linhas 4 – 5 inicializam os vetores de falsos positivos e falsos negativos. A classificação de cada amostra é executada nas Linhas 6 – 11, atualizando os vetores de falsos positivos e falsos negativos. Amostras classificadas incorretamente são armazenadas na lista  $M$  (Linha 11). A Linha 12 salva a instância atual do classificador e calcula a acurácia da iteração  $I$ , a qual é armazenada na curva de aprendizagem  $\mathcal{L}$  (Linha 13). O laço nas Linhas 14 – 17 troca as amostras de  $Z_2$  classificadas incorretamente por amostras de  $Z_1$  selecionadas aleatoriamente, sob as determinadas restrições mencionadas anteriormente.

Uma versão diferente deste algoritmo de aprendizagem foi apresentada por Papa et al. [56]. O algoritmo calculava o grau de relevância de cada amostra em  $Z_1$ , baseado no número de classificações corretas e incorretas em  $Z_2$  que envolviam aquela amostra no caminho ótimo. O grau de relevância de cada amostra era calculado subtraindo o seu número de classificações incorretas do seu número de classificações corretas. Amostras com graus negativos eram consideradas irrelevantes, sendo selecionadas para o processo de troca. A idéia era eliminar outliers de  $Z_1$ , visto que essas amostras são geralmente irrelevantes. O algoritmo de aprendizagem descrito acima (Algoritmo 6) é mais simples e rápido que o anterior (o que considera o grau de relevância de cada nó em  $Z_1$ ), ou seja, não precisa calcular o grau de relevância de todos os nós em  $Z_1$ . Ele tem se mostrado superior em todas as bases de dados testadas, exceto para a base Brain [16].

# Capítulo 5

## Resultados Experimentais

Esta seção tem por objetivo apresentar as bases de dados (Seção 5.1), descritores (Seção 5.2) e os experimentos realizados (Seções 5.3, 5.4 e 5.5), os quais comparam o classificadores baseados em OPF com SVM, ANN-MLP, KNN e BC com relação à sua acurácia e eficiência (tempo computacional). Os classificadores baseados em floresta de caminhos ótimos adotados nos experimentos foram aqueles que fazem uso do aprendizado supervisionado, descritos nas Seções 4.2.1 e 4.2.2.

### 5.1 Bases de dados

A Tabela 5.1 mostra as 11 bases de dados utilizadas nos experimentos, com diversos tipos de amostras. A base MPEG-7 [53], por exemplo, utiliza formas (contornos) de imagens (Figura 5.1), COREL [18] utiliza imagens coloridas (Figura 5.2), apresentadas aqui em escala monocromática, e a base Brodatz [10] utiliza imagens de textura (Figura 5.3). Estas bases de dados permitem avaliar o desempenho dos classificadores de acordo com descritores de forma, cor e textura, respectivamente. As demais bases já possuem seus próprios vetores de atributos: WBC (*Wisconsin Breast Cancer*), IS (*Image Segmentation*) e LR (*Letter Recognition*) [1]; Brain [16]; e as bases Cone-torus, Saturno, Pétalas e Boat [45]. A base Brain utiliza voxels para as amostras de substância branca e cinzenta de imagens de ressonância magnética do cérebro humano, com vários níveis de ruído para produzir outliers. Os atributos são valor mínimo, máximo e intensidade de uma pequena vizinhança 3D de cada voxel. As quatro últimas bases de dados utilizam as coordenadas  $(x, y)$  de pontos 2D como seus atributos (Figura 5.4).

Código	Nome	número de objetos	número de classes
B <sub>1</sub>	MPEG-7	1400	70
B <sub>2</sub>	COREL	1607	49
B <sub>3</sub>	Brodatz	208	13
B <sub>4</sub>	WBC	699	2
B <sub>5</sub>	IS	2310	7
B <sub>6</sub>	LR	5000	20
B <sub>7</sub>	Brain	1578	2
B <sub>8</sub>	Cone-torus	400	3
B <sub>9</sub>	Saturno	200	2
B <sub>10</sub>	Pétalas	100	4
B <sub>11</sub>	Boat	100	3

Tabela 5.1: Descrição das bases de dados.

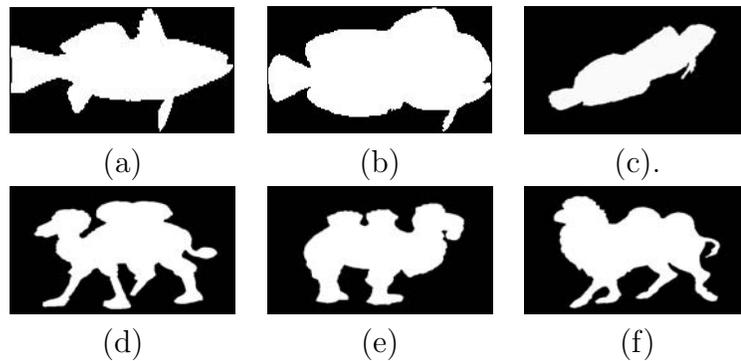


Figura 5.1: Exemplos de formas da base MPEG-7 das classes (a)-(c) peixes e (d)-(f) camelos.

## 5.2 Descritores

A Tabela 5.2 mostra 10 diferentes possibilidades que combinam extração dos atributos  $v$  e função de distância  $d$  para formar os descritores  $(v, d)$ . Alguns descritores foram implementados para formas (D<sub>1</sub>-D<sub>5</sub>), cor (D<sub>6</sub>-D<sub>7</sub>) e imagens de textura (D<sub>8</sub>). Os descritores D<sub>1</sub>, D<sub>2</sub> e D<sub>3</sub> utilizam os coeficientes de Fourier (*Fourier Coefficients* - FC) [61], os Invariantes de Momento (*Moment Invariants* - MI) [37] e as dimensões fractais multiescala (*Multiscale Fractal Dimensions* - MSF) [70] como atributos de forma, respectivamente, e a norma Euclidiana (L2) como função de distância. Os descritores D<sub>4</sub> e D<sub>5</sub> calculam três medidas estatísticas, chamadas de BAS (*Beam Angle Statistics*), para cada amostra do contorno de um objeto [3]. Eles utilizam as métricas L2 e OCS (*Optimal Correspon-*

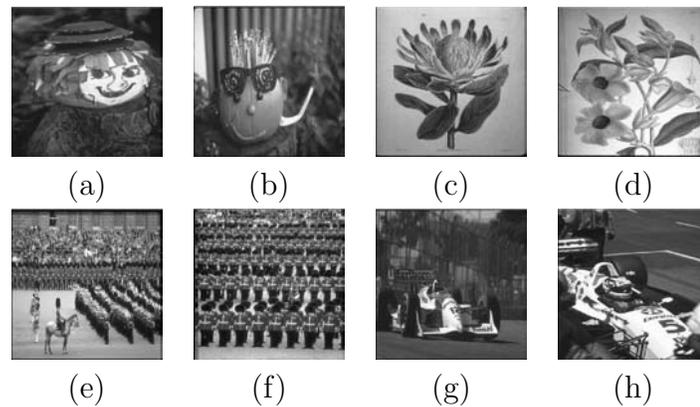


Figura 5.2: Exemplos de imagens da base Corel das classes (a)-(b) abóbora, (c)-(d) flores, (e)-(f) exército britânico, e (g)-(h) carros de corrida.

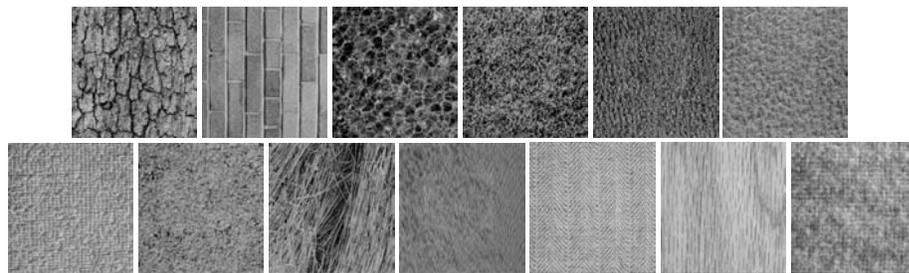


Figura 5.3: Imagens de textura da base Brodatz. Cada imagem, da esquerda para a direita e de cima para baixo, representa uma classe: casca de árvore, tijolo, bolhas, grama, couro, pele de porco, fibra vegetal, areia, palha, água, tecido, madeira e lã.

*dence of String Subsequences*) [72], respectivamente, para comparação entre seus vetores de atributos, evidenciando a importância de funções de distâncias mais específicas como o OCS, por exemplo.

A comparação entre os descritores  $D_1$  a  $D_5$  utilizando o mesmo classificador ilustra suas habilidades em representar as formas de uma determinada base de dados. O descritor  $D_6$  classifica os pixels em regiões de borda/interior e calcula os histogramas de cor de cada uma dessas regiões [66], utilizando a métrica L1 entre o logaritmo desses histogramas (dLog) como função de distância. Imagens coloridas são também representadas pelos seus histogramas de cor (CHIST) [67] e comparadas com a métrica L2 no descritor  $D_7$ . O descritor  $D_8$  utiliza a decomposição piramidal por *wavelets* para criar os vetores de textura (TEX), os quais são comparados por um algoritmo de extração de texturas invariante a rotação (*Rotation-invariant Texture Matching* - RIM) [50]. O descritor  $D_9$  representa todos os vetores de atributos (OWN) já disponíveis nas bases de dados  $B_4$  a  $B_7$  e  $D_{10}$  representa os pontos no espaço 2D (XY) como sendo os atributos das bases  $B_8$  a  $B_{11}$

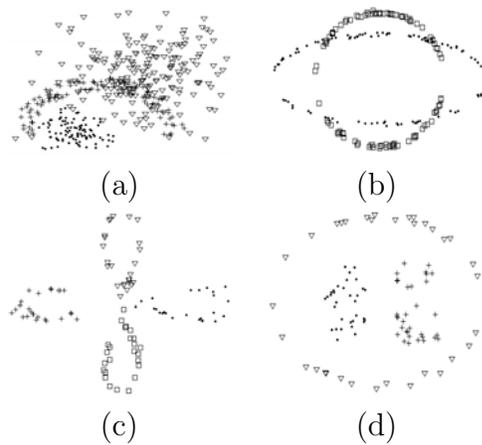


Figura 5.4: Bases de dados de pontos 2D: (a) Cone-torus, (b) Saturno, (c) Pétalas, e (d) Boat.

(Tabela 5.1). A função de distância L2 é utilizada para essas bases de dados. Finalmente, as combinações entre bases de dados e descritores é sumarizada na Tabela 5.3.

Alguns classificadores assumem o espaço de atributos Euclidiano implicitamente, como ANN-MLP, e outros possuem funções de distâncias embutidas em seu modelo, como as funções de base radial (*Radial Basis Function* - RBF) presentes nos kernels da SVM. Quando a função de distância utilizada é a L2, usamos como RBF para o algoritmo SVM

$$K(s, t) = \exp^{-\gamma \|\vec{v}(s) - \vec{v}(t)\|^2}, \quad (5.1)$$

onde  $s$  e  $t$  são as duas amostras (uma delas é o vetor de suporte) e  $\vec{v}(s)$  and  $\vec{v}(t)$  são seus respectivos vetores de atributos. A constante  $\gamma$  é encontrada por otimização de parâmetros [13]. No caso de funções de distância  $d$  mais complexas, observamos uma considerável melhora no desempenho do classificador SVM quando trocamos seu kernel RBF por

$$K'(s, t) = \exp^{-\gamma d^2(s, t)}. \quad (5.2)$$

Assim sendo,  $K'$  foi utilizado em todos os experimentos envolvendo SVM e funções de distâncias mais complexas  $d$  e  $K$  foi usado para experimentos que utilizavam a métrica L2.

A metodologia acima foi também adotada para os classificadores restantes utilizados. Embora a distância Euclidiana seja adotada como padrão para o classificador  $k$ -NN, utilizamos outras funções de distância mais complexas com o intuito de identificar os  $k$  vizinhos mais próximos. A mesma situação aplica-se ao classificador Bayesiano (BC), o qual calcula as densidades das amostras utilizando um kernel gaussiano (em casos nos

Descriptor Code	Feature extraction algorithm ( $v$ )	Distance function ( $d$ )
D <sub>1</sub>	FC	L2
D <sub>2</sub>	MI	L2
D <sub>3</sub>	MSF	L2
D <sub>4</sub>	BAS	L2
D <sub>5</sub>	BAS	OCS
D <sub>6</sub>	BIC	dLog
D <sub>7</sub>	CHIST	L2
D <sub>8</sub>	TEX	RIM
D <sub>9</sub>	OWN	L2
D <sub>10</sub>	XY	L2

Tabela 5.2: Descritores utilizados nos experimentos.

quais a fdp é gaussiana) com distância Euclideana, a qual foi substituída por outras funções de distância quando necessárias.

Os experimentos descritos nas próximas seções avaliaram a acurácia no conjunto de teste  $Z_3$  e o tempo computacional de cada classificador, os baseados em OPF, SVM, ANN-MLP,  $k$ -NN e BC, para cada par base de dados e descritor. A Seção 5.3 apresenta os valores de acurácia do treinamento em  $Z_1$  e teste em  $Z_3$ . Os resultados do treinamento em  $Z_1$  com aprendizado através dos erros em  $Z_2$ , e teste em  $Z_3$  são apresentados na Seção 5.4. O tempo computacional médio de cada classificador para o treinamento e classificação é dividido pelo número de amostras, e apresentado na Seção 5.5. Para os experimentos realizados sem aprendizado em  $Z_2$  (Seção 5.3), as bases de dados foram divididas em duas partes: um conjunto de treinamento  $Z_1$  com 50% das amostras e um conjunto de testes  $Z_3$  com 50% das amostras. Para os experimentos realizados com aprendizado em  $Z_2$  (Seção 5.4), as bases de dados foram divididas em três partes: um conjunto de treinamento  $Z_1$  com 30% das amostras, um conjunto de avaliação com  $Z_2$  com 20% das amostras e um conjunto de teste  $Z_3$  com 50% das amostras, as quais foram aleatoriamente selecionadas.

Cada experimento foi repetido 10 vezes com diferentes conjuntos  $Z_1$  e  $Z_3$  para o caso do treinamento em  $Z_1$  e teste em  $Z_3$ , e 10 vezes com diferentes conjuntos  $Z_1$ ,  $Z_2$  e  $Z_3$  para o caso do treinamento em  $Z_1$ , aprendizado em  $Z_2$  e teste em  $Z_3$ ; com o intuito de calcular a acurácia média e o seu desvio padrão, bem como o valor médio do coeficiente Kappa [15]. Para facilitar a notação dos classificadores baseados em floresta de caminhos ótimos, denotamos  $\text{OPF}_{cpl}$  como sendo o classificador descrito na Seção 4.2.1 e  $\text{OPF}_{k-nn}$  como sendo o classificador descrito na Seção 4.2.2.

Código da base de dados	Código do descritor
B <sub>1</sub>	D <sub>1</sub> ,D <sub>2</sub> ,D <sub>3</sub> ,D <sub>4</sub> ,D <sub>5</sub>
B <sub>2</sub>	D <sub>6</sub> ,D <sub>7</sub>
B <sub>3</sub>	D <sub>8</sub>
B <sub>4</sub>	D <sub>9</sub>
B <sub>5</sub>	D <sub>9</sub>
B <sub>6</sub>	D <sub>9</sub>
B <sub>7</sub>	D <sub>9</sub>
B <sub>8</sub>	D <sub>10</sub>
B <sub>9</sub>	D <sub>10</sub>
B <sub>10</sub>	D <sub>10</sub>
B <sub>11</sub>	D <sub>10</sub>

Tabela 5.3: Bases de dados e seus respectivos descritores utilizados nos experimentos.

Os experimentos a seguir (Seções 5.3 e 5.4) serão apresentados em duas partes cada um, sendo uma delas com uma tabela comparativa entre os classificadores similares (OPF<sub>cpl</sub> x  $k$ -NN x OPF <sub>$k$ -nn</sub> x BC) e outra entre os classificadores mais populares (OPF<sub>cpl</sub> x OPF <sub>$k$ -nn</sub> x SVM x ANN-MLP).

### 5.3 Resultados obtidos diretamente no conjunto de teste

Os resultados disponíveis nas Tabelas 5.4 e 5.5 são apresentados na seguinte forma:  $x \pm y(z)[k]$ , onde  $x$ ,  $y$ ,  $z$  e  $k$  são a acurácia média, seu desvio padrão, o valor do coeficiente Kappa médio [15] e o valor do melhor  $k$  obtido, respectivamente, sendo este último apresentado somente no caso do classificador  $k$ -NN. Valores de Kappa abaixo de 0.8 indicam a dificuldade em classificar algumas bases de dados utilizando seus respectivos descritores. Já bons descritores tendem a melhor separar as classes no espaço de atributos, reduzindo a sobreposição e facilitando o processo de classificação dos dados. As Tabelas 5.4 e 5.5 apresentam, respectivamente, uma comparação entre os algoritmos ditos similares e populares.

Nota-se que os valores de acurácia dos classificadores baseados em OPF e SVM foram claramente melhores que os valores obtidos pela ANN-MLP,  $k$ -NN e BC. Contudo, OPF e SVM apresentaram desempenho geral equivalente, sendo um melhor que o outro dependendo do caso. Os valores de exatidão ao longo das linhas de B<sub>1</sub> (D<sub>4</sub>) e B<sub>1</sub> (D<sub>5</sub>) indicam a importância da utilização de funções distância mais específicas, exceto para a ANN-MLP,

Base(Descriptor)	Classificadores			
	OPF <sub>cpl</sub>	<i>k</i> -NN	OPF <sub><i>k</i>-nn</sub>	BC
<i>B</i> <sub>1</sub> ( <i>D</i> <sub>1</sub> )	<b>71.71±0.8(0.49)</b>	59.38±1.1(0.17)[1]	69.87±0.7(0.39)	64.10±0.6(0.28)
<i>B</i> <sub>1</sub> ( <i>D</i> <sub>2</sub> )	79.48±0.9(0.59)	72.04±1.0(0.64)[1]	<b>80.25±1.6(0.60)</b>	73.24±0.7(0.46)
<i>B</i> <sub>1</sub> ( <i>D</i> <sub>3</sub> )	74.81±0.4(0.49)	60.16±0.4(0.19)[1]	<b>75.95±0.9(0.51)</b>	73.08±0.5(0.46)
<i>B</i> <sub>1</sub> ( <i>D</i> <sub>4</sub> )	86.14±0.3(0.72)	66.55±0.3(0.67)[1]	<b>87.37±0.3(0.74)</b>	81.32±0.6(0.62)
<i>B</i> <sub>1</sub> ( <i>D</i> <sub>5</sub> )	<b>95.72±0.9(0.89)</b>	50.70±0.5(0.31)[1]	94.45±0.6(0.87)	93.82±0.3(0.87)
<i>B</i> <sub>2</sub> ( <i>D</i> <sub>6</sub> )	<b>86.74±0.1(0.75)</b>	82.83±0.1(0.70)[1]	80.01±0.2(0.64)	71.28±0.1(0.60)
<i>B</i> <sub>2</sub> ( <i>D</i> <sub>7</sub> )	<b>80.25±0.1(0.63)</b>	78.03±0.1(0.61)[1]	79.42±0.2(0.62)	56.61±0.1(0.51)
<i>B</i> <sub>3</sub> ( <i>D</i> <sub>8</sub> )	83.43±1.6(0.66)	84.52±1.7(0.80)[1]	<b>88.85±2.3(0.77)</b>	80.31±2.0(0.60)
<i>B</i> <sub>4</sub> ( <i>D</i> <sub>9</sub> )	93.87±1.2(0.88)	91.85±0.6(0.81)[3]	<b>93.89±1.2(0.88)</b>	91.77±2.4(0.85)
<i>B</i> <sub>5</sub> ( <i>D</i> <sub>9</sub> )	74.37±0.1(0.48)	65.89±0.1(0.41)[2]	<b>79.37±0.1(0.68)</b>	77.69±0.1(0.63)
<i>B</i> <sub>6</sub> ( <i>D</i> <sub>9</sub> )	<b>90.35±0.1(0.80)</b>	87.20±0.1(0.79)[2]	90.03±0.1(0.80)	88.27±0.1(0.79)
<i>B</i> <sub>7</sub> ( <i>D</i> <sub>9</sub> )	90.53±0.2(0.81)	86.39±0.3(0.73)[1]	<b>90.70±0.5(0.81)</b>	90.19±0.5(0.80)
<i>B</i> <sub>8</sub> ( <i>D</i> <sub>10</sub> )	<b>87.29±1.7(0.71)</b>	81.34±1.5(0.65)[7]	85.76±2.2(0.70)	75.70±2.4(0.56)
<i>B</i> <sub>9</sub> ( <i>D</i> <sub>10</sub> )	88.10±0.3(0.76)	81.90±0.2(0.62)[1]	<b>88.20±0.3(0.76)</b>	81.70±0.4(0.63)
<i>B</i> <sub>10</sub> ( <i>D</i> <sub>10</sub> )	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)[21]</b>	<b>100.0±0.0(1.0)</b>	81.70±4.0(0.63)
<i>B</i> <sub>11</sub> ( <i>D</i> <sub>10</sub> )	96.76±0.1(0.93)	93.19±0.01(0.89)[1]	<b>97.05±0.1(0.94)</b>	89.55±4.5(0.79)

Tabela 5.4: Resultados  $x \pm y(z)$  em  $Z_3$  sem a utilização de  $Z_2$  entre os algoritmos similares:  $x$  - acurácia média,  $y$  - desvio padrão das acurácias e  $z$  - Kappa médio. As melhores acurácias são evidenciadas em negrito.

a qual não utiliza funções de distância. Os resultados ao longo das cinco primeiras linhas também indicam a importância do descritor para a base de dados  $B_1$ . A Figura 5.5 mostra os resultados obtidos diretamente no conjunto de testes para os classificadores baseados em OPF, SVM, ANN,  $k$ -NN e BC.

## 5.4 Resultados obtidos no conjunto de teste após aprendizado com conjunto de avaliação

Com o intuito de avaliar a habilidade dos classificadores em aprender a partir dos seus erros em  $Z_2$  sem aumentar o tamanho do conjunto de treinamento  $Z_1$ , executamos o Algoritmo 6 com  $I = 10$  iterações. Novamente, os classificadores baseados em OPF e SVM apresentaram desempenhos equivalentes, superiores aos resultados obtidos por ANN-MLP,  $k$ -NN e BC (Tabelas 5.6 e 5.7).

Entretanto, observamos que os algoritmos baseados em OPF possuem um aprendizado

Base(Descriptor)	Classificadores			
	OPF <sub>cpl</sub>	OPF <sub>k-nn</sub>	SVM	ANN-MLP
$B_1 (D_1)$	<b>71.71±0.8(0.49)</b>	69.87±0.7(0.39)	70.07±0.7(0.40)	57.28±44.3(0.14)
$B_1 (D_2)$	79.48±0.9(0.59)	80.25±1.6(0.60)	<b>82.15±1.1(0.64)</b>	71.48±26.5(0.46)
$B_1 (D_3)$	74.81±0.4(0.49)	<b>75.95±0.9(0.51)</b>	74.49±0.9(0.50)	62.98±39.8(0.25)
$B_1 (D_4)$	86.14±0.3(0.72)	<b>87.37±0.3(0.74)</b>	87.05±0.6(0.75)	77.99±34.8(0.57)
$B_1 (D_5)$	<b>95.72±0.9(0.89)</b>	94.45±0.6(0.87)	94.92±0.3(0.88)	76.29±4.9(0.55)
$B_2 (D_6)$	86.74±0.1(0.75)	80.01±0.2(0.64)	<b>90.65±0.1(0.83)</b>	83.07±11.2(0.64)
$B_2 (D_7)$	80.25±0.1(0.63)	79.42±0.2(0.62)	<b>83.37±0.1(0.70)</b>	80.07±10.1(0.61)
$B_3 (D_8)$	83.43±1.6(0.66)	<b>88.85±2.3(0.77)</b>	84.27±1.6(0.68)	86.97±21.0(0.73)
$B_4 (D_9)$	93.87±1.2(0.88)	93.89±1.2(0.88)	<b>95.46±0.01(0.90)</b>	92.83±0.02(0.86)
$B_5 (D_9)$	74.37±0.1(0.48)	<b>79.37±0.1(0.68)</b>	78.35±1.0(0.59)	73.35±20.3(0.68)
$B_6 (D_9)$	90.35±0.1(0.80)	90.03±0.1(0.80)	<b>93.35±0.1(0.90)</b>	84.72±1.6(0.73)
$B_7 (D_9)$	90.53±1.7(0.81)	90.70±0.5(0.81)	<b>93.86±2.2(0.88)</b>	92.94±9.7(0.85)
$B_8 (D_{10})$	<b>87.29±1.7(0.71)</b>	85.76±2.2(0.70)	85.54±2.4(0.71)	85.33±24.37(0.69)
$B_9 (D_{10})$	88.10±0.3(0.76)	<b>88.20±0.3(0.76)</b>	86.90±0.5(0.73)	83.60±5.1(0.67)
$B_{10} (D_{10})$	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>
$B_{11} (D_{10})$	96.76±0.1(0.93)	97.05±0.2(0.94)	<b>99.55±0.1(0.99)</b>	97.20±3.0(0.94)

Tabela 5.5: Resultados  $x \pm y(z)$  em  $Z_3$  sem a utilização de  $Z_2$  entre os algoritmos populares:  $x$  - acurácia média,  $y$  - desvio padrão das acurácias e  $z$  - Kappa médio. As melhores acurácias são evidenciadas em negrito.

mais rápido que os demais classificadores, como pode ser observado pela Figura 5.6. Isto implica que o tempo de execução do algoritmo de aprendizado pode ser sensivelmente diminuído, bastando parar sua execução quando a acurácia atingir o valor máximo ou quando a mesma estabilizar. A Figura 5.7 mostra os resultados obtidos com aprendizado em  $Z_2$  para os classificadores baseados em OPF, SVM, ANN,  $k$ -NN e BC.

## 5.5 Resultados em eficiência

A Tabela 5.8 apresenta o tempo médio de execução em segundos dividido pelo número de elementos que cada classificador utiliza no seu processo de treinamento e classificação para cada base de dados e descriptor. Nota-se que os algoritmos baseados em OPF são mais eficientes que os demais algoritmos. O classificador  $OPF_{cpl}$  foi, em média, 1.07 vezes mais rápido que  $OPF_{knn}$ , 55.01 vezes mais rápido que SVM, 72.50 vezes mais rápido que ANN-MLP, e 1.38 e 1.4663 vezes mais rápido que os classificadores  $k$ -NN e BC, respectivamente.

Base(Descriptor)	Classificadores			
	OPF <sub>cpl</sub>	<i>k</i> -NN	OPF <sub><i>k</i>-nn</sub>	BC
$B_1 (D_1)$	73.82±0.6(0.47)	59.38±1.1(0.17)[1]	<b>75.94±1.0(0.51)</b>	69.92±0.4(0.40)
$B_1 (D_2)$	<b>81.20±0.8(0.60)</b>	72.88±1.0(0.44)[3]	81.03±1.3(0.62)	79.53±0.3(0.59)
$B_1 (D_3)$	76.72±0.9(0.53)	61.48±1.2(0.26)[3]	<b>76.88±0.9(0.53)</b>	75.49±0.81(0.50)
$B_1 (D_4)$	87.24±0.2(0.74)	67.14±0.6(0.68)[1]	<b>87.65±0.3(0.75)</b>	87.00±(0.5)0.74
$B_1 (D_5)$	<b>96.08±0.6(0.90)</b>	50.70±0.1(0.31)[1]	95.75±0.4(0.89)	95.71±0.5(0.91)
$B_2 (D_6)$	86.90±0.1(0.76)	82.83±0.1(0.73)[1]	<b>87.87±0.2(0.78)</b>	75.74±0.2(0.62)
$B_2 (D_7)$	80.29±0.1(0.63)	79.73±0.1(0.61)[1]	<b>80.67±0.2(0.64)</b>	61.23±0.1(0.31)
$B_3 (D_8)$	88.54±2.4(0.77)	86.26±2.1(0.70)[1]	<b>90.41±2.5(0.80)</b>	87.70±1.8(0.75)
$B_4 (D_9)$	94.17±0.9(0.89)	91.26±0.9(0.81)[9]	<b>94.75±1.0(0.90)</b>	91.54±1.6(0.85)
$B_5 (D_9)$	75.90±0.1(0.51)	67.06±0.1(0.24)[2]	<b>79.90±0.1(0.70)</b>	77.99±0.1(0.65)
$B_6 (D_9)$	92.07±0.1(0.84)	89.54±0.1(0.81)[9]	<b>92.40±0.1(0.84)</b>	92.34±0.1(0.84)
$B_7 (D_9)$	<b>91.53±0.5(0.83)</b>	88.99±0.5(0.76)[1]	91.45±0.7(0.83)	91.29±1.5(0.81)
$B_8 (D_{10})$	<b>88.38±1.4(0.72)</b>	83.43±1.6(0.68)[11]	86.28±2.0(0.71)	75.11±0.04(0.55)
$B_9 (D_{10})$	88.70±0.2(0.776)	83.90±0.2(0.63)[1]	<b>89.30±0.2(0.78)</b>	86.40±6.3(0.72)
$B_{10} (D_{10})$	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b> [5]	<b>100.0±0.0(1.0)</b>	98.46±1.8(0.97)
$B_{11} (D_{10})$	97.35±0.2(0.94)	94.81±0.3(0.86)[1]	<b>97.64±0.2(0.95)</b>	91.17±5.6(0.82)

Tabela 5.6: Resultados  $x \pm y(z)$  em  $Z_3$  com aprendizado em  $Z_2$  entre os algoritmos similares:  $x$  - acurácia média,  $y$  - desvio padrão das acurácias e  $z$  - Kappa médio. As melhores acurácias são evidenciadas em negrito.

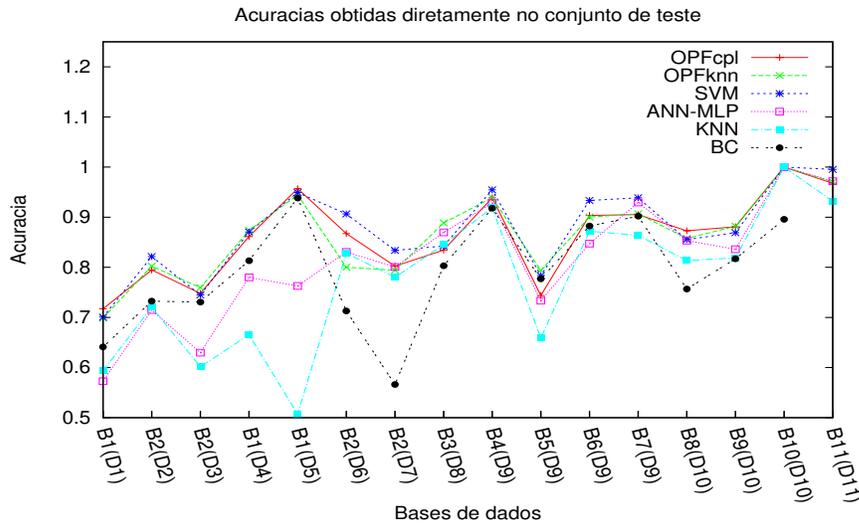


Figura 5.5: Resultados obtidos diretamente no conjunto de teste para os classificadores OPF<sub>cpl</sub>, OPF<sub>knn</sub>, SVM, ANN-MLP, *k*-NN e BC.

Base(Descriptor)	Classificadores			
	OPF <sub>cpl</sub>	OPF <sub>k-nn</sub>	SVM	ANN-MLP
$B_1 (D_1)$	73.82±0.6(0.47)	<b>75.94±1.0(0.51)</b>	74.42±0.4(0.48)	61.39±6.4(0.22)
$B_1 (D_2)$	81.20±0.8(0.60)	81.03±1.3(0.62)	<b>82.03±0.7(0.64)</b>	75.06±0.3(0.50)
$B_1 (D_3)$	76.72±0.9(0.53)	<b>76.88±0.9(0.53)</b>	76.52±0.9(0.53)	64.76±18.5(0.29)
$B_1 (D_4)$	87.24±0.2(0.74)	<b>87.65±0.3(0.75)</b>	87.18±0.4(0.73)	79.23±22.2(0.58)
$B_1 (D_5)$	<b>96.08±0.6(0.90)</b>	95.75±0.4(0.89)	95.87±0.1(0.90)	78.65±0.9(0.57)
$B_2 (D_6)$	86.90±0.1(0.76)	87.87±0.2(0.78)	<b>90.80±0.2(0.84)</b>	85.70±0.2(0.75)
$B_2 (D_7)$	80.29±0.1(0.63)	80.67±0.2(0.64)	83.66±0.1(0.71)	<b>84.70±0.1(0.79)</b>
$B_3 (D_8)$	88.54±2.4(0.77)	90.41±2.5(0.80)	84.37±1.5(0.68)	<b>92.29±16.6(0.84)</b>
$B_4 (D_9)$	94.17±0.9(0.89)	94.75±1.0(0.90)	<b>96.07±1.1(0.91)</b>	93.26±19.1(0.87)
$B_5 (D_9)$	75.90±0.1(0.51)	<b>79.90±0.1(0.70)</b>	78.65±0.1(0.57)	74.35±0.1(0.70)
$B_6 (D_9)$	92.07±0.1(0.84)	92.40±0.1(0.84)	<b>94.31±0.1(0.93)</b>	87.72±0.1(0.79)
$B_7 (D_9)$	91.53±0.5(0.83)	91.45±0.7(0.83)	<b>94.00±0.7(0.88)</b>	93.73±6.9(0.87)
$B_8 (D_{10})$	<b>88.38±1.4(0.72)</b>	86.28±2.0(0.71)	87.95±3.0(0.74)	85.58±17.4(0.70)
$B_9 (D_{10})$	88.70±0.2(0.776)	<b>89.30±0.2(0.78)</b>	<b>89.30±0.3(0.78)</b>	88.10±0.4(0.76)
$B_{10} (D_{10})$	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>	<b>100.0±0.0(1.0)</b>
$B_{11} (D_{10})$	97.35±0.2(0.94)	97.64±0.2(0.95)	<b>99.85±0.1(0.99)</b>	98.23±0.2(0.96)

Tabela 5.7: Resultados  $x \pm y(z)$  em  $Z_3$  com aprendizado em  $Z_2$  entre os algoritmos populares:  $x$  - acurácia média,  $y$  - desvio padrão das acurácias e  $z$  - Kappa médio. As melhores acurácias são evidenciadas em negrito.

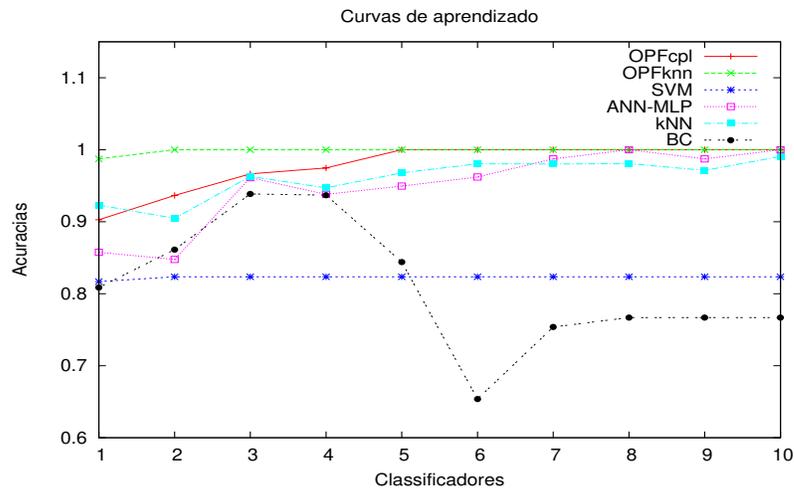


Figura 5.6: Curvas de aprendizado para os classificadores OPF<sub>cpl</sub>, OPF<sub>k-nn</sub>, SVM, ANN-MLP e  $k$ -NN, usando o Algoritmo 6 com  $I = 10$  iterações. Os valores de acurácia foram obtidos sobre o conjunto avaliação  $Z_2$  na base de dados Cone-torus.

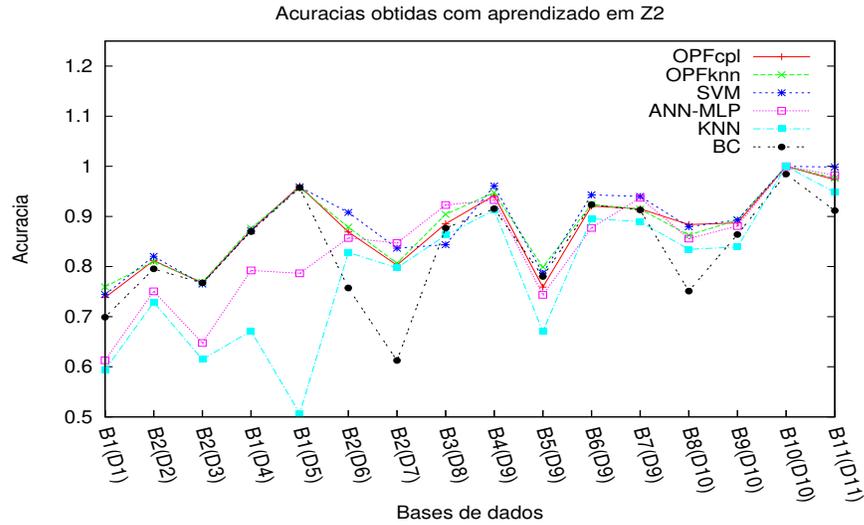


Figura 5.7: Resultados obtidos com aprendizado em  $Z_2$  para os classificadores  $OPF_{cpl}$ ,  $OPF_{knn}$ , SVM, ANN-MLP,  $k$ -NN e BC.

Base(Descriptor)	Classificadores					
	OPF <sub>cpl</sub>	OPF <sub>k-nn</sub>	SVM	ANN-MLP	$k$ -NN	BC
$B_1 (D_1)$	<b>0.0052</b>	0.0113	1.304	0.8973	0.0450	0.0850
$B_1 (D_2)$	<b>0.0010</b>	0.0020	0.0599	0.3453	0.0034	0.0031
$B_1 (D_3)$	<b>0.0012</b>	<b>0.0012</b>	0.0742	0.3603	0.0038	0.0040
$B_1 (D_4)$	<b>0.0019</b>	0.0038	0.2859	0.5043	0.0126	0.0138
$B_1 (D_5)$	5.8400	5.6400	7.3926	<b>0.5031</b>	5.8432	5.8425
$B_2 (D_6)$	<b>0.0185</b>	0.0202	0.1813	0.2553	0.0199	0.0199
$B_2 (D_7)$	<b>0.0010</b>	0.0011	0.0997	0.2491	0.0254	0.0011
$B_3 (D_8)$	<b>0.2163</b>	0.2164	0.2333	0.2891	0.2164	0.2173
$B_4 (D_9)$	<b>0.0019</b>	<b>0.0019</b>	0.0091	0.01505	0.0042	0.0099
$B_5 (D_9)$	<b>0.0018</b>	0.0058	0.0693	0.400	0.0010	0.0004
$B_6 (D_9)$	<b>0.0012</b>	0.0014	0.0320	0.0800	0.0017	0.0016
$B_7 (D_9)$	<b>0.0011</b>	<b>0.0011</b>	0.1662	3.5625	0.01960	0.0020
$B_8 (D_{10})$	<b>0.0010</b>	0.0040	0.1281	1.8540	0.0011	0.0030
$B_9 (D_{10})$	<b>0.0011</b>	<b>0.0011</b>	0.1445	0.3828	0.0002	0.0020
$B_{10} (D_{10})$	<b>0.0018</b>	<b>0.0018</b>	0.0078	0.0020	0.0003	<b>0.0018</b>
$B_{11} (D_{10})$	<b>0.0019</b>	<b>0.0019</b>	0.0594	0.0039	<b>0.0019</b>	<b>0.0019</b>

Tabela 5.8: Tempo de execução médio em segundos para os processos de treinamento e classificação dividido pelo número de amostras.

# Capítulo 6

## Conclusão e trabalhos futuros

O presente trabalho teve, como objetivo, apresentar novas abordagens na área de pesquisa em reconhecimento supervisionado de padrões: classificadores baseados em floresta de caminhos ótimos, os quais computam uma floresta de caminhos ótimos no conjunto de treinamento e classificam as amostras com o rótulo de sua raiz mais fortemente conexa nesta árvore. Tais raízes são protótipos de todas as classes, os quais podem ser obtidos por diferentes heurísticas.

Os classificadores baseados em OPF modelam as amostras do conjunto de dados como sendo os nós de um grafo, e são conectados de acordo com uma relação de adjacência pré-estabelecida. Através dos elementos protótipos, caminhos de custo ótimo são oferecidos para as demais amostras (nós) do conjunto de dados iniciando, assim, um processo competitivo entre os protótipos, os quais tentam conquistar os outros elementos oferecendo caminhos de menor custo, que podem ser computados através de funções de valor de caminho também pré-estabelecidas. Assim, a presente tese de doutorado abre um campo para pesquisa em classificadores baseados em floresta de caminhos ótimos, onde novas funções de custo de caminho, relações de adjacência e heurísticas para encontrar protótipos podem ser combinadas de diferentes maneiras com o intuito de gerar novos classificadores.

Dois classificadores baseados em OPF foram apresentados: o que faz uso do grafo completo ( $\text{OPF}_{cpl}$ ) e outro que faz uso de um grafo  $k$ -nn ( $\text{OPF}_{k-nn}$ ). O classificador  $\text{OPF}_{cpl}$  modela as amostras como sendo os nós de um grafo completo e estima os protótipos nas regiões de fronteira entre as classes. As arestas do grafo são ponderadas com a distância entre os vetores de atributos dos seus respectivos nós. Para obter os protótipos, o algoritmo computa uma MST no grafo de treinamento e marca como protótipos os elementos conexos na MST de diferentes classes. Assim, tais protótipos iniciam um processo de conquista no grafo de treinamento, oferecendo caminhos de menor custo para os demais nós. A função de valor de caminho utilizada é a  $f_{max}$ , a qual minimiza os máximos dos

valores dos caminhos entre os protótipos e os outros nós do grafo. Este processo gera uma floresta de caminhos ótimos no grafo de treinamento, onde cada árvore de caminhos ótimos é enraizada pelo seu respectivo protótipo. O rótulo das amostras de cada árvore é mesmo de sua respectiva raiz. A fase de teste consiste, basicamente, em inserir um elemento do conjunto de teste nessa floresta de caminhos ótimos, conectá-lo a todos os nós e repetir o procedimento acima.

O classificador  $\text{OPF}_{k\text{-nn}}$  modela as amostras como sendo os nós de um grafo  $k\text{-nn}$ , onde  $k$  é obtido como sendo o valor que minimiza o erro de classificação no conjunto de treinamento. O  $\text{OPF}_{k\text{-nn}}$  estima os protótipos nas regiões de maior densidade de nós, os quais novamente tentam oferecer caminhos de custo ótimo, originando uma floresta de caminhos ótimos no grafo de treinamento. As arestas são novamente ponderadas com a distância entre os vetores de atributos dos respectivos nós. Entretanto, nesta abordagem, os nós são também ponderados por valores de densidade de probabilidade, os quais são computados usando o peso dos arcos. A função de valor de caminho utilizada é a  $f_{\min}$ , a qual maximiza os mínimos dos valores dos caminhos ótimos entre os protótipos e os demais nós do grafo. Esta função pode ser entendida como sendo uma função dual de  $f_{\max}$ .

Apresentamos também um algoritmo de aprendizado, o qual permite a um classificador aprender com seus próprios erros em um conjunto de avaliação e aumentar sua acurácia no conjunto de testes, sem aumentar o tamanho do conjunto de treinamento. A motivação para tal abordagem vem da utilização dos classificadores como parte de sistemas especialistas, os quais podem, com o passar do tempo, retreinar os classificadores com seus novos erros, sem aumentar o tamanho do conjunto de treinamento.

Comparamos os classificadores baseados em OPF com outros classificadores supervisionados de padrões amplamente adotados na literatura: SVM, ANN-MLP,  $k\text{-NN}$  e BC. Utilizamos várias bases de dados e descritores, com e sem utilização do algoritmo de aprendizado. Com relação a valores de acurácia, os classificadores baseados em OPF obtiveram resultados similares aos encontrados pelo SVM, e superiores aos resultados de ANN-MLP,  $k\text{-NN}$  e BC. Entretanto, os algoritmos  $\text{OPF}_{cpl}$  e  $\text{OPF}_{knn}$  foram mais computacionalmente eficientes que os demais classificadores comparados na seção experimental. Os classificadores baseados em OPF também apresentam algumas vantagens teóricas, tais como (i) ausência de parâmetros (no caso do  $\text{OPF}_{cpl}$ ), (ii) trabalham com problemas multiclases sem modificações ou extensões e (iii) não assumem forma e/ou separabilidade das classes.

Criamos também a biblioteca LibOPF [57], um conjunto de funções implementadas na linguagem C com o intuito de disponibilizar o código do OPF para qualquer pessoa que se interesse em estudá-lo. Utilizamos esse pacote de implementação para a realização dos experimentos descritos anteriormente.

Atualmente, estamos investigando a utilização de Programação Genética (*Genetic Pro-*

*gramming* - GP) para estimação do peso do arco nos classificadores baseados em OPF. Podemos combinar as distâncias de descritores de forma, cor e textura com GP, por exemplo, e utilizar o resultado para ponderar os arcos. Obtivemos resultados promissores com a utilização de GP para combinar distâncias entre descritores no contexto de recuperação de imagens por conteúdo [71], fato este que nos motivou a utilizar GP também no contexto de classificação de imagens.

# Trabalhos publicados

## Conferências

- João Paulo Papa, Alexandre Xavier Falcão, Paulo André Vechiatto de Miranda, Celso Tetsuo Nagase Suzuki e Nelson Delfino d'Ávila Mascarenhas. **Design of Robust Pattern Classifiers based on Optimum-path forests.** *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pp. 337–348, MCT/INPE, Rio de Janeiro - Brazil, 2007.
- Javier Alexander Montoya-Zegarra, João Paulo Papa, Neucimar Jerônimo Leite, Ricardo da Silva Torres e Alexandre Xavier Falcão. **Rotation-invariant Texture Recognition.** *III International Symposium on Visual Computing*, pp. 193–204, vol. 4842, Springer, Lake Tahoe - EUA, 2007.
- André Augusto Spadotto, José Carlos Pereira, Rodrigo Capobianco Guido, João Paulo Papa, Alexandre Xavier Falcão, Ana Rita Gatto, Paula Cristina Cola e Arthur Oscar Shelp. **Oropharyngeal Dysphagia Identification Using Wavelets and Optimum Path Forest.** *III International Symposium on Communications, Control and Signal Processing*. St. Julians - Malta, pp. 735-740, ISBN: 978-1-4244-1688-2, 2008.
- João Paulo Papa, Alexandre Xavier Falcão, Celso Tetsuo Nagase Suzuki e Nelson Delfino d'Ávila Mascarenhas. **A Discrete Approach for Supervised Pattern Recognition.** *XII International Workshop on Combinatorial Image Analysis*, pp. 136–147, vol. 4958, Springer Berlin/Heidelberg, Buffalo - EUA, 2008.
- João Paulo Papa, André Augusto Spadotto, Alexandre Xavier Falcão e José Carlos Pereira. **Optimum Path Forest Classifier Applied to Laryngeal Pathology Detection.** *XV International Conference on Systems, Signals and Image Processing*, pp. 249–252, Publishing House STU, Bratislava - Eslováquia, 2008.
- João Paulo Papa e Alexandre Xavier Falcão. **A New Variant of the Optimum-Path Forest Classifier.** *IV International Symposium on Visual Computing*, Las Vegas - EUA, 2008. (aceito para publicação)

- Javier Alexander Montoya-Zegarra, João Paulo Papa, Neucimar Jerônimo Leite, Ricardo da Silva Torres e Alexandre Xavier Falcão. **Novel approaches for exclusive and continuous fingerprint classification.** *III Pacific-Rim Symposium on Image and Video Technology*, Tóquio - Japão, 2009. (aceito para publicação)

### Periódicos

- Javier Alexander Montoya-Zegarra, João Paulo Papa, Neucimar Jerônimo Leite, Ricardo da Silva Torres e Alexandre Xavier Falcão. **Learning How to Extract Rotation-Invariant and Scale-Invariant Features from Texture Images.** *Journal on Advances in Signal Processing*, 2008 (no prelo).
- Ricardo da Silva Torres, Alexandre Xavier Falcão, Marcos André Gonçalves, João Paulo Papa, Baoping Zhang, Weiguo Fan e Edward Fox. **A Genetic Programming Framework for Content-based Image Retrieval.** *Pattern Recognition*, 2008. (no prelo)

### Patentes

- Alexandre Xavier Falcão, Celso Tetsuo Nagase Suzuki, Jancarlo Ferreira Gomes, João Paulo Papa, Luiz Cândido de Souza Dias, Sumie Hoshino-Shimizu. **Sistema para Diagnóstico de Parasitos Intestinais por Análise Computadorizada de Imagens.** *Protocolo INPI - PI0605465-0 (pedido solicitado)*, Instituto de Computação, Universidade Estadual de Campinas.
- Alexandre Xavier Falcão, Celso Tetsuo Nagase Suzuki, Jancarlo Ferreira Gomes, João Paulo Papa, Luiz Cândido de Souza Dias, Sumie Hoshino-Shimizu. **A system for diagnosing intestinal parasites by computerized image analysis.** *Protocolo PCT - PCT/BR2007/000272 (pedido solicitado)*, Instituto de Computação, Universidade Estadual de Campinas.
- Alexandre Xavier Falcão, Celso Tetsuo Nagase Suzuki, Jancarlo Ferreira Gomes, Luiz Cândido de Souza Dias, Sumie Hoshino-Shimizu, João Paulo Papa. **Sistema para Diagnóstico de Parasitos Intestinais por Análise Computadorizada de Imagens e Uso de Referido Sistema.** *Protocolo PCT - PCT/BR2007/000272 (pedido solicitado)*, Instituto de Computação, Universidade Estadual de Campinas.

### Relatórios técnicos

- João Paulo Papa, Alexandre Xavier Falcão, Paulo André Vechiatto de Miranda, Celso Tetsuo Nagase Suzuki e Nelson Delfino d'Ávila Mascarenhas. **A New Pattern Classifier based on Optimum Path Forest.** Relatório Técnico IC-07-13, Instituto de Computação, Universidade Estadual de Campinas.

- João Paulo Papa, Alexandre Xavier Falcão e Celso Tetsuo Nagase Suzuki. **Supervised Pattern Classification based on Optimum-Path Forest**. Relatório Técnico IC-08-20, Instituto de Computação, Universidade Estadual de Campinas.

# Referências Bibliográficas

- [1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.
- [2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
- [3] N. Arica and F.T.Y. Vural. BAS: A Perceptual Shape Descriptor Based on the Beam Angle Statistics. *Pattern Recognition Letters*, 24(9-10):1627–1639, June 2003.
- [4] R. Audigier, R.A. Lotufo, and A.X. Falcão. 3D visualization to assist iterative object definition from medical images. *Computerized Medical Imaging and Graphics*, 30(4):217–230, 2006.
- [5] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th the International Conference on Machine Learning*, 2002.
- [6] F.P.G. Bergo, A.X. Falcão, P.A.V. Miranda, and L.M. Rocha. Automatic image segmentation by tree pruning. *Journal of Mathematical Imaging and Vision*, 29(2–3):141–162, 2007.
- [7] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing, Real-Time Edge and Motion Detection*, 1979.
- [8] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA, 1998. ACM Press.
- [9] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.
- [10] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, New York, 1966.

- [11] F.A.M. Cappabianco, A.X. Falcão, and L.M. Rocha. Clustering by optimum path forest and its application to automatic gm/wm classification in mr-t1 images of the brain. In *ISBI*, pages 428–431, 2008.
- [12] G. Castellano, R.A. Lotufo, A.X. Falcão, and F. Cendes. Characterization of the human cortex in MR images through the image foresting transform. In *Proceedings of IEEE International Conference on Image Processing*, pages 357–360, Sep 2003.
- [13] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at url <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [14] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, Aug 1995.
- [15] J. Cohen. A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement*, volume 20, pages 37–46, 1960.
- [16] D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.
- [17] R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 23, New York, NY, USA, 2004. ACM Press.
- [18] Corel Corporation. *Corel stock photo images*. <http://www.corel.com>.
- [19] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- [20] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [21] K. Duan and S.S. Keerthi. Which is the best multiclass svm method? an empirical study. *Multiple Classifier Systems*, pages 278–285, 2005.
- [22] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000.
- [23] A.X. Falcão and F.P.G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Transactions on Medical Imaging*, 23(9):1100–1108, 2004.

- [24] A.X. Falcão, L.F. Costa, and B.S. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, Apr 2002.
- [25] A.X. Falcão, L.F. Costa, and B.S. da Cunha. Erratum to Multiscale Skeletons by Image Foresting Transform and its Applications to Neuromorphometry: [Pattern Recognition 35(7) (2002) 1571-1582]. *Pattern Recognition.*, 36(12):3013, Dec 2003.
- [26] A.X. Falcão, B.S. da Cunha, and R.A. Lotufo. Design of connected operators using the image foresting transform. In *Proceedings of SPIE on Medical Imaging*, volume 4322, pages 468–479, Feb 2001.
- [27] A.X. Falcão, J. Stolfi, and R.A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.
- [28] A.X. Falcão, C.T.N. Suzuki, J.F. Gomes, L. Candido, S. Shimizu, and J.P. Papa. Sistema para diagnóstico de parasitos intestinais por análise computadorizada de imagens e uso de referido sistema. Protocolo PI018080046387 (pedido solicitado).
- [29] A.X. Falcão, C.T.N. Suzuki, J.F. Gomes, J.P. Papa, L. Candido, and S. Shimizu. Sistema para diagnóstico de parasitos intestinais por análise computadorizada de imagens. Protocolo INPI - PI0605465-0 (pedido solicitado).
- [30] A.X. Falcão, C.T.N. Suzuki, J.F. Gomes, J.P. Papa, L. Candido, and S. Shimizu. A system for diagnosing intestinal parasites by computerized image analysis. Protocolo PCT - PCT/BR2007/000272 (pedido solicitado).
- [31] A.X. Falcão and J.K. Udupa. A 3D generalization of user-steered live wire segmentation. *Medical Imaging Analysis*, 4(4):389–402, Dec 2000.
- [32] A.X. Falcão, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. *IEEE Transactions on Medical Imaging*, 19(1):55–62, Jan 2000.
- [33] A.X. Falcão, J.K. Udupa, S. Samarasekera, and B.E. Hirsch. User-steered image boundary segmentation. In *Proceedings of SPIE on Medical Imaging*, volume 2710, pages 278–288, Feb. 1996.
- [34] K. Fukunaga and P.M. Narendra. A branch and bound algorithms for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24(7):750–753, 1975.

- [35] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 187–194, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.
- [36] S. Haykin. *Neural Networks: A comprehensive foundation*. Prentice-Hall, 1998.
- [37] M.K. Hu. Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.
- [38] S. C. Huang and Y. F. Huang. Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):47–55, 1991.
- [39] T.-M. Huang, V. Kecman, and I. Kopriva. *Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-supervised, and Unsupervised Learning (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [40] L.J. Hubert. Some applications of graph theory to clustering. *Psychometrika*, 39(3):283–309, 1974.
- [41] A.K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- [42] A.K. Jain, R. P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
- [43] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [44] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.
- [45] L. Kuncheva. Artificial data. *School of Informatics, University of Wales*, 1996. [http://www.informatics.bangor.ac.uk/~kuncheva/activities/artificial\\_data.htm](http://www.informatics.bangor.ac.uk/~kuncheva/activities/artificial_data.htm).
- [46] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
- [47] C.-J. Lin. Formulations of support vector machines: A note from an optimization point of view. *Neural Computation*, 13(2):307–317, 2001.

- [48] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 341–350. Kluwer, Jun 2000.
- [49] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley*, pages 281–297. University of California Press, 1967.
- [50] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, and A.X. Falcão. Rotation-invariant texture recognition. In *3rd International Symposium on Visual Computing*, volume Part II, LNCS 4842, pages 193–204, Lake Tahoe, Nevada, CA, USA, Nov 2007. Springer.
- [51] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, and A.X. Falcão. Learning how to extract rotation-invariant and scale-invariant features from texture images. In *Journal on Advances in Signal Processing*, volume 2008, pages 1–16, 2008.
- [52] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, and A.X. Falcão. Novel approaches for exclusive and continuous fingerprint classification. In *III Pacific-Rim Symposium on Image and Video Technology*, 2009. (aceito para publicação).
- [53] MPEG-7. Mpeg-7: The generic multimedia content description standard, part 1. *IEEE MultiMedia*, 09(2):78–87, 2002.
- [54] S. Nissen. *Implementation of a fast artificial neural network library (FANN)*, 2003. Department of Computer Science University of Copenhagen (DIKU). Software available at <http://leenissen.dk/fann/>.
- [55] N. Panda, E.Y. Chang, and G. Wu. Concept boundary detection for speeding up svms. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 681–688, New York, NY, USA, 2006. ACM Press.
- [56] J. P. Papa, A.X. Falcão, P. A. V. Miranda, C. T. N. Suzuki, and N. D. A. Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. In *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 337–348. MCT/INPE, 2007.
- [57] João Paulo Papa, Celso Tetsuo Nagase Suzuki, and Alexandre Xavier Falcão. *LibOPF: A library for the design of optimum-path forest classifiers*, 2008. Software version 1.0 available at <http://www.ic.unicamp.br/~afalcao/LibOPF>.
- [58] J.P. Papa and A.X. Falcão. A new variant of the optimum-path forest classifier. In *4th International Symposium on Visual Computing*, 2008. (aceito para publicação).

- [59] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, and N.D.A. Mascarenhas. A discrete approach for supervised pattern recognition. In *12th International Workshop on Combinatorial Image Analysis*, volume 4958, pages 136–147. Springer Berlin/Heidelberg, 2008.
- [60] J.P. Papa, A.A. Spadotto, A.X. Falcão, and J.C. Pereira. Optimum path forest classifier applied to laryngeal pathology detection. In *15th International Conference on Systems, Signals and Image Processing*, pages 249–252. Publishing House STU, Bratislava, 2008.
- [61] E. Persoon and K. Fu. Shape Discrimination Using Fourier Descriptors. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(3):170–178, 1977.
- [62] L. Reyzin and R.E. Schapire. How boosting the margin can also boost classifier complexity. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 753–760, New York, NY, USA, 2006. ACM Press.
- [63] L. M. Rocha, A. X. Falcão, and L. G. P. Meloni. A robust extension of the mean shift algorithm using optimum path forest. In *8th Intl. Workshop on Combinatorial Image Analysis*, pages 29–38, Buffalo-NY, USA, 2008. RPS (ISBN 978-981-08-0228-8).
- [64] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.
- [65] A.A. Spadotto, J.C. Pereira, R.C. Guido, J.P. Papa, A.X. Falcão, A.R. Gatto, P.C. Cola, and A.O. Shelp. Oropharyngeal dysphagia identification using wavelets and optimum path forest. In *Proceedings of the 3th IEEE International Symposium on Communications, Control and Signal Processing*, pages 735–740, 2008. ISBN: 978-1-4244-1688-2.
- [66] R.O. Stehling, M.A. Nascimento, and A.X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 102–109, New York, NY, USA, 2002. ACM Press.
- [67] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [68] B. Tang and D. Mazzone. Multiclass reduced-set support vector machines. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 921–928, New York, NY, USA, 2006. ACM Press.

- [69] R.S. Torres and A.X. Falcão. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3–13, Jan 2007.
- [70] R.S. Torres, A.X. Falcão, and L.F. Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163–1174, 2004.
- [71] R.S. Torres, A.X. Falcão, M.A. Gonçalves, J.P. Papa, B. Zhang, W. Fan, and E.A. Fox. A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):283–292, 2009.
- [72] Y.P. Wang and T. Pavlids. Optimal Correspondence of String Subsequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1080–1087, 1990.
- [73] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.
- [74] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, Jan. 1971.
- [75] T. Zhang and F.J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1191–1198, 2000.
- [76] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.