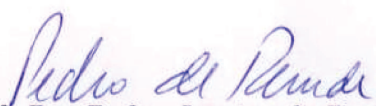


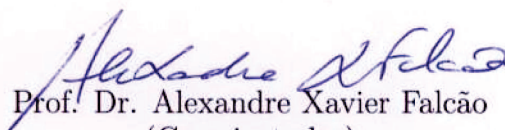
Novos Algoritmos de Aprendizado para Classificação de Padrões Utilizando Floresta de Caminhos Ótimos

César Christian Castelo
Fernández

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por César Christian Castelo Fernández e aprovada pela Banca Examinadora.

Campinas, 11 de Maio de 2011.


Prof. Dr. Pedro Jussieu de Rezende
(Orientador)


Prof. Dr. Alexandre Xavier Falcão
(Co-orientador)

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

**FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP**
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Castelo Fernández, César Christian

C277n Novos algoritmos de aprendizado para classificação de padrões utilizando floresta de caminhos ótimos/César Christian Castelo Fernández-- Campinas, [S.P. : s.n.], 2011.

Orientador : Pedro Jussieu de Rezende; Alexandre Xavier Falcão.

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Reconhecimento de padrões. 2.Aprendizado do computador.
3.Teoría dos grafos. 4.Outliers (Estatística). 5.Processamento de
imagens. 6.Visão por computador. I. Rezende, Pedro Jussieu de.
II.Falcão, Alexandre Xavier. III. Universidade Estadual de Campinas.
Instituto de Computação. IV. Título.

Título em inglês: New learning algorithms for pattern classification using optimum-path forest

Palavras-chave em inglês (Keywords): 1.Pattern recognition. 2.Machine learning. 3.Graph theory. 4.Outliers (Statistics). 5.Image processing. 6.Machine vision.

Área de concentração: Ciência da Computação

Titulação: Mestre em Ciência da Computação

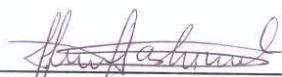
Banca examinadora: Prof. Dr. Pedro Jussieu de Rezende (IC – UNICAMP)
Prof. Dr. Hélio Pedrini (IC– UNICAMP)
Prof. Dr. Ronaldo Fumio Hashimoto (IME - USP)

Data da defesa: 11/05/2011

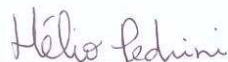
Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO


Dissertação Defendida e Aprovada em 11 de maio de 2011, pela Banca examinadora composta pelos Professores Doutores:



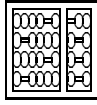
Prof. Dr. Ronaldo Fumio Hashimoto
IME / USP



Prof. Dr. Hélio Pedrini
IC / UNICAMP



Prof. Dr. Pedro Jussieu de Rezende
IC / UNICAMP



Novos Algoritmos de Aprendizado para Classificação de Padrões Utilizando Floresta de Caminhos Ótimos

César Christian Castelo Fernández¹

Maio de 2011

Banca Examinadora:

- Prof. Dr. Pedro Jussieu de Rezende (Orientador)
- Prof. Dr. Helio Pedrini
Instituto de Computação, Universidade Estadual de Campinas (IC-UNICAMP)
- Prof. Dr. Ronaldo Fumio Hashimoto
Instituto de Matemática e Estatística, Universidade de São Paulo (IME-USP)
- Prof. Dr. Ricardo da Silva Torres (Suplente)
Instituto de Computação, Universidade Estadual de Campinas (IC-UNICAMP)
- Prof. Dr. Aparecido Nilceu Marana (Suplente)
Departamento de Ciência da Computação, Universidade Estadual Paulista - Baurú
(DCC-UNESP-BAURÚ)

¹Bolsista CAPES, Processo # 01-P-04388/2010, de março de 2009 a fevereiro de 2011; Bolsita PED-C (IC-Unicamp) de março de 2010 a julho de 2010.

Resumo

O Reconhecimento de Padrões pode ser definido como a capacidade de identificar a classe de algum objeto dentre um dado conjunto de classes, baseando-se na informação fornecida por amostras conhecidas (conjunto de treinamento). Nesta dissertação, o foco de estudo é o paradigma de classificação supervisionada, no qual se conhece a classe de todas as amostras utilizadas para o projeto do classificador. Especificamente, estuda-se o Classificador baseado em Floresta de Caminhos Ótimos (*Optimum-Path Forest - OPF*) e propõem-se três novos algoritmos de aprendizado, os quais representam melhorias em comparação com o Classificador OPF tradicional.

Primeiramente, é desenvolvida uma metodologia simples, porém efetiva, para detecção de outliers no conjunto de treinamento. O método visa uma melhoria na acurácia do Classificador OPF tradicional através da troca desses outliers por novas amostras do conjunto de avaliação e sua exclusão do processo de aprendizagem. Os outliers são detectados computando uma penalidade para cada amostra baseada nos seus acertos e erros na classificação, o qual pode ser medido através do número de falsos positivos/negativos e verdadeiros positivos/negativos obtidos por cada amostra. O método obteve uma melhoria na acurácia em comparação com o OPF tradicional, com apenas um pequeno aumento no tempo de treinamento.

Em seguida, é proposto um aprimoramento ao primeiro algoritmo, que permite detectar com maior precisão os outliers presentes na base de dados. Neste caso, utiliza-se a informação de falsos positivos/negativos e verdadeiros positivos/negativos de cada amostra para explorar intrinsecamente as relações de adjacência de cada amostra e determinar se é outlier. Uma inovação do método é que não existe necessidade de se computar explicitamente tal adjacência, como é feito nas técnicas tradicionais, o qual pode ser inviável para grandes bases de dados. O método obteve uma boa taxa de detecção de outliers e um tempo de treinamento muito baixo em vista do tamanho das bases de dados utilizadas.

Finalmente, é abordado o problema de se selecionar um número tão pequeno quanto possível de amostras de treinamento e se obter a maior acurácia possível sobre o conjunto de teste. Propõe-se uma metodologia que se inicia com um pequeno conjunto de treinamento e, através da classificação de um conjunto bem maior de avaliação, aprende quais amostras são as mais representativas para o conjunto de treinamento. Os resultados mostram que é possível obter uma melhor acurácia que o Classificador OPF tradicional ao custo de um pequeno incremento no tempo de treinamento, mantendo, no entanto, o conjunto de treinamento menor que o conjunto inicial, o que significa um tempo de teste reduzido.

Abstract

Pattern recognition can be defined as the capacity of identifying the class of an object among a given set of classes, based on the information provided by known samples (training set). In this dissertation, the focus is on the supervised classification approach, for which we are given the classes of all the samples used in the design of the classifier. Specifically, the Optimum-Path Forest Classifier (OPF) is studied and three new learning algorithms are proposed, which represent improvements to the traditional OPF classifier.

First of all, a simple yet effective methodology is developed for the detection of outliers in a training set. This method aims at improving OPF's accuracy through the swapping of outliers for new samples from the evaluating set and their exclusion from the learning process itself. Outliers are detected by computing a penalty for each sample based on its classification-hits and -misses, which can be measured through the number of false positive/negatives and true positives/negatives obtained by each sample. The method achieved an accuracy improvement over the traditional OPF, with just a slight increment in the training time.

An improvement to the first algorithm is proposed, allowing for a more precise detection of outliers present in the dataset. In this case, the information on the number of false positive/negatives and true positives/negatives of each sample is used to explore the adjacency relations of each sample and determine whether it is an outlier. The method's merit is that there is no need of explicitly computing an actual vicinity, as the traditional techniques do, which could be infeasible for large datasets. The method achieves a good outlier detection rate and a very low training time, considering the size of the datasets.

Finally, the problem of choosing a small number of training samples while achieving a high accuracy in the testing set is addressed. We propose a methodology which starts with a small training set and, through the classification of a much larger evaluating set, it learns which are the most representative samples for the training set. The results show that it is possible to achieve higher accuracy than the traditional OPF's at the cost of a slight increment in the training time, preserving, however, a smaller training set than the original one, leading to a lower testing time.

Ao meu pai, Hugo César Castelo Del Carpio (in memoriam).

A mi más grande tesoro

Cuando nací fuiste el hombre más feliz del mundo porque esperaste mucho mi llegada ...

Cuando empecé a crecer me cuidaste como a tu más grande tesoro ...

Fuiste tú quien me enseñó lo linda que es la vida y que siempre debemos estar felices, como lo demostraste a cada momento con tu cálida sonrisa y tu alegría tan particular en las fiestas ...

Fuiste tú quien me dotó de toda su inteligencia, su amor al trabajo y su responsabilidad ...

Con tus consejos me enseñaste a conducirme en la vida y a ser un hombre de bien ...

Si hoy yo estoy realizando mis sueños es gracias a ti, a tu inagotable amor y tus incansables sacrificios porque nunca me falte nada ...

A cada segundo me demostraste cuánto me amaste, incluso estando lejos, con tal que yo pueda tener todo lo que necesitaba ...

Hoy que ya te fuiste, te guardo siempre en mi corazón porque tú eres mi más grande tesoro ...

Te agradezco todo tu amor y tus cuidados y ahora que estás al lado de nuestro Señor sé que estás conmigo más que nunca ...

GRACIAS POR TODO PAPITO, TE AMARÉ SIEMPRE!!!!

Agradecimentos

A Deus, por ter me permitido chegar até o Brasil para fazer o meu mestrado e por iluminar meus passos durante o tempo que morei aqui, guiando-me para conhecer pessoas de bom coração que sempre me ajudaram em tudo o que precisei.

Ao meu orientador, Prof. Pedro J. de Rezende, por ter acreditado em mim desde o começo do mestrado; pela sua paciência e perseverança nas longas discussões, nas quais adquiri inúmeros conhecimentos; pelos valiosos conselhos ao longo da minha formação, que me tornaram não apenas um pesquisador, mas também uma melhor pessoa; pela exigência que sempre teve com o meu trabalho, pois isso gerou uma dedicação cada vez maior da minha parte e principalmente, pela amizade adquirida.

Ao meu co-orientador, Prof. Alexandre X. Falcão, por ter acreditado em mim quando aceitou a co-orientação; pela sua paciência em escutar cada uma das minhas dúvidas; pelo seu entusiasmo e alegria em todas nossas conversas, nas quais consegui aprender muito com a sua grande experiência; pela sua orientação nos temas de pesquisa e principalmente, pela amizade adquirida.

Ao Prof. Helio Pedrini, pela sua amizade, sua amabilidade, seus conselhos sobre o meu futuro, sua disposição para conversar sempre comigo, e pelas dicas nas disciplinas e no meu Exame de Qualificação.

Ao meu pai Hugo César Castelo Del Carpio (*in memoriam*) que, com certeza, estaria muito orgulhoso de mim neste momento, por estar cumprindo o seu grande desejo de que eu estudasse aqui no Brasil. Eu sei que você me acompanhou durante este tempo, pai, nos momentos bons e também nos difíceis, torcendo para que eu lograsse meus objetivos. **ESSA É PARA VOCÊ, PAI!**

À minha mãe, Himelda Fernández de Castelo, e minha irmã, Carmen Castelo Fernández, que sempre torceram por mim e, mesmo estando tão longe, sempre estiveram aí quando precisei de alguém a quem falar. À minha mãe, que nunca se cansou de escutar, nas longas conversas por telefone, as minhas alegrias, tristezas, preocupações, lembranças e planos para o futuro; pela sua confiança de que lograria as minhas metas; pela sua preocupação comigo; pelas suas constantes palavras de apoio; pela sua alegria com as minhas conquistas e pelo seu amor inesgotável ... em outras palavras, por ser a melhor mãe do

mundo! À minha irmã, pelo seu grande amor e confiança de que eu conseguiria vencer todos os obstáculos; por todos os planos que fazia quando eu voltava para o Perú; pela alegria e entusiasmo com que esperava a minha chegada; por planejar minhas recepções e despedidas em cada viagem e por cuidar e ajudar à nossa mãe enquanto eu fiquei aqui. Muito obrigado às duas! EU AMO MUITO VOCÊS! (*Las quiero mucho! LPM!*).

A toda minha família, em especial aos meus avós Julio (*in memoriam*), Raúl (*in memoriam*), Carmen e Angelica, aos meus tios Norma, Gladys, Teresa, Carlos e Sonia, aos meus primos, às minhas primas e ao meu cunhado, porque sempre torceram por mim desde o Perú e por terem cuidado da minha mãe e da minha irmã.

Aos amigos Paulo, Mariângela, Maíra, Naara e Tainá que sempre me deram o seu carinho e me fizeram sentir em casa cada vez que estava em São Paulo. Muito obrigado por acolher a este peruano dentro da sua família!

Ao amigo e colega João Paulo Papa, pela sua ajuda ao longo de todo o meu mestrado, proporcionando-me valiosas dicas para o meu Exame de Qualificação, na elaboração dos testes, na escrita dos artigos e, com certeza, pela amizade adquirida.

Aos meus amigos e colegas do Laboratório de Informática Visual (LIV), e aos amigos do Instituto de Computação (IC) que conheci ao longo desses dois anos, agradeço pela sua amizade, sua ajuda e suas dicas.

Aos meus amigos da República, pela sua amizade e camaradagem, pelas interessantes conversas nos momentos vagos, pelos bons momentos compartilhados e pela sua companhia nesses dois anos. Obrigado pela oportunidade de conhecer grandes pessoas como vocês e por terem feito a convivência tão agradável!

Aos amigos Sheila, Ricardo e Marcelo pela oportunidade de ter grandes amigos como vocês, pelas constantes palavras de apoio, as longas conversas e o apoio em tudo. À Sheila pela sua grande ajuda no começo do meu mestrado, quando cheguei, não conhecia ninguém e não tinha onde ficar. Ao Ricardo, pelas suas orações, seus conselhos e suas palavras de alento em todo momento. Ao Marcelo, pela sua camaradagem e sua alegria tão particular.

Ao Governo Brasileiro através da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), pelo apoio financeiro durante estes 24 meses, o qual fez possível a realização do meu trabalho.

À Universidade Estadual de Campinas (Unicamp) por me acolher nas suas aulas e aos professores e funcionários do Instituto de Computação (IC-Unicamp), Daniel, Fernando, Ademilson e Flávio, pelo apoio constante e informação brindados durante esses dois anos.

A todos que colaboraram, direta ou indiretamente, com o meu trabalho, meu muito obrigado!

Sumário

Resumo	vii
Abstract	ix
Agradecimentos	xiii
1 Introdução	1
1.1 Motivação do Trabalho	3
1.2 Problemas de interesse	3
1.2.1 Detecção de Outliers	4
1.2.2 Construção de um Conjunto de Treinamento adequado	4
1.3 Proposta e Contribuições Principais	4
1.3.1 Detecção de Outliers	4
1.3.2 Construção de um Conjunto de Treinamento adequado	5
1.4 Organização do Trabalho	5
2 Conceitos sobre Reconhecimento de Padrões	7
2.1 Etapas no Projeto do Classificador	7
2.2 Separabilidade de Classes	8
2.3 Abordagens para Classificação de Padrões	8
2.3.1 Classificação Não-Supervisionada	8
2.3.2 Classificação Semi-Supervisionada	9
2.3.3 Classificação Supervisionada	9
3 Revisão de Classificadores	11
3.1 Classificador Bayesiano	11
3.1.1 Definições	11
3.1.2 Estimativas de Funções de Densidade de Probabilidade	13
3.2 k Vizinhos mais Próximos	14
3.3 Redes Neurais Artificiais	16
3.3.1 Perceptrons	17
3.3.2 Treinamento por Retropropagação	18
3.4 Máquinas de Vetores de Suporte	19
3.4.1 Classificação Binária	19

3.4.2	Classificação por Hiperplano Ótimo	20
3.5	Floresta de Caminhos Ótimos	22
3.5.1	Definições	22
3.5.2	Treinamento	23
3.5.3	Classificação	23
4	Improving the Accuracy of the Optimum-Path Forest Supervised Classifier for Large Datasets	27
4.1	Introduction	28
4.2	Optimum-path forest classifier	29
4.3	Optimum-Path Forest Classifier with Outliers Detection: OPF-OD	30
4.4	Experiments	32
4.4.1	Datasets used	32
4.4.2	Sizes of Z_1 and Z_2	33
4.4.3	Results	33
4.5	Conclusions	34
	Bibliography	35
	Comentários	36
	Domínio das Bases de Dados	36
	Gráfico dos Resultados	36
5	A Novel Outlier Detection Method based on the Optimum-Path Forest Supervised Classifier	37
5.1	Introduction	38
5.2	Optimum-path forest classifier	39
5.2.1	Training	41
5.2.2	Classification	41
5.2.3	Learning	42
5.3	OPF-based Outlier Detection: OPF-OD	43
5.3.1	Defining the outliers in the training set	44
5.3.2	Swapping outliers and wrongfully classified samples from the evaluating set	45
5.3.3	Proposed algorithm	45
5.4	Experiments	46
5.4.1	Definition of outlier	46
5.4.2	Datasets	47
5.4.3	Efficiency in training	47
5.4.4	Efficiency of the outlier detection task	48
5.5	Conclusions	49
	Bibliography	50
	Comentários	53
	Determinação dinâmica do limiar de Grau de Isolamento	53
	Gráfico dos Resultados	53
	Resultados da base Gamma	53

Correção: Tempo de processamento	54
6 Towards Building an Ideal Training Set for Supervised Pattern Classification using Optimum-Path Forest	55
6.1 Introduction	56
6.2 Optimum-path forest classifier	57
6.2.1 Selection of Prototypes	59
6.2.2 Classification of Evaluating Samples	59
6.2.3 Learning Algorithms for OPF	59
6.3 Smart Growth of the Training Set	60
6.3.1 Initial Sampling for Z_1	61
6.3.2 Selecting the most Suitable Samples	62
6.3.3 Defining the Stopping Conditions	63
6.4 Experiments	63
6.4.1 Datasets	63
6.4.2 Experimental methodology	64
6.4.3 Stopping conditions	64
6.4.4 Evaluation of the algorithm on Z_2	66
6.4.5 Evaluation of the algorithm on Z_3	68
6.4.6 Final results	68
6.5 Conclusions	70
Bibliography	71
Comentários	73
Critérios de parada adicionais	73
Medida de acurácia	73
Gráfico dos Resultados	73
Correção: Tempo de teste	73
Redução do conjunto do treinamento	74
7 Trabalhos futuros	75
8 Considerações Finais	77
Bibliografia	79

Lista de Tabelas

4.1	Resultados do método OPF-OD.	34
5.1	Bases de dados utilizadas nos testes do método OPF-OD aprimorado. . . .	47
5.2	Resultados do método OPF-OD aprimorado.	49
6.1	Bases de dados utilizadas nos testes do método OPF-SGTS.	64
6.2	Resultados do método OPF-SGTS (Acurácia em Z_3).	69
6.3	Resultados do método OPF-SGTS (Tempo de treinamento).	69
6.4	Estatísticas da fase de aprendizado para o método OPF-SGTS.	70

Lista de Figuras

1.1	Etapas envolvidas no projeto de um sistema de classificação de padrões. . .	2
2.1	Casos de separabilidade entre duas classes representadas no espaço 2D. . .	8
2.2	Processo seguido na classificação supervisionada, incluindo treinamento, aprendizagem e classificação de novas amostras (teste).	10
3.1	Exemplo de Funções de Densidade de Probabilidade considerando as classes ω_1 e ω_2 . O erro na classificação é representado pela região em cinza. . . .	13
3.2	Exemplo de Probabilidades Condicionais para as PDFs da Figura 3.1, considerando $P(\omega_1) = 2/3$ e $P(\omega_2) = 1/3$	14
3.3	Exemplo de um Diagrama de Voronoi considerando $N = 30$ em um espaço 2D e usando distância euclideana.	15
3.4	Exemplo de uma Rede Neural Artificial com três camadas e duas saídas. .	16
3.5	Exemplo de classificação utilizando Máquinas de Vetores de Suporte. Os vetores de suporte definem os hiperplanos H_1 e H_2	21
3.6	Exemplo de classificação utilizando Floresta de Caminhos Ótimos. Apresenta-se o grafo completo, a árvore MST, os protótipos escolhidos e a Floresta de Caminhos Ótimos gerada.	25
4.1	Resultados do método OPF-OD, considerando a Acurácia na classificação de Z_3 , utilizando diferentes tamanhos para Z_1	33
4.2	Resultados do método OPF-OD, considerando Tempo de treinamento e Tempo de classificação, utilizando diferentes tamanhos para Z_1	34
5.1	Exemplo de uma Floresta de Caminhos Ótimos gerada a partir de um grafo completo.	41
5.2	Processo de escolha de protótipos para o método OPF utilizando uma Árvore Geradora Mínima.	42
5.3	Processo de classificação do método OPF.	42
5.4	Tempo de treinamento para OPF, OPF-OD e SVM usando a base de dados Shuttle com diferentes tamanhos para $ Z_1 $	48
5.5	Resultados do método OPF-OD aprimorado para a base de dados Shuttle, apresentado-se a Curva ROC e o melhor método dinâmico.	50

6.1	Demonstração da validade da utilização do limiar de diferença simétrica. Apresenta-se o processo de aprendizado para o método OPF-SGTS, utilizando a base de dados Gamma.	65
6.2	Determinação de um limiar de diferença simétrica adequado para o método OPF-SGTS, utilizando a base de dados Gamma.	66
6.3	Processo de aprendizado para o método OPF-SGTS, utilizando a base de dados Gamma.	67
6.4	Processo de aprendizado para o método OPF-SGTS, utilizando a base de dados Gamma e medindo a acurácia em Z_3 (em lugar de Z_2).	68

Capítulo 1

Introdução

Reconhecimento de Padrões pode ser definido como a capacidade de tomar uma decisão com base na *categoria (classe)* atribuída a um objeto ou conjunto de objetos, genericamente chamados de *padrões* [14].

O processo de identificar a categoria do objeto em questão pode ser uma tarefa trivial para um ser humano porque é parte da nossa vida cotidiana. Alguns exemplos são: reconhecer a face de uma pessoa, entender as palavras que escutamos, ler um texto, identificar qual é a chave adequada que estamos procurando, etc. No entanto, esta tarefa pode ser extremamente complexa para ser realizada automaticamente por um sistema de computador e, por isso, é um campo de pesquisa muito ativo, pois oferece soluções a problemas em muitas áreas do conhecimento, tais como medicina, biologia, parasitologia, química, física e bioinformática, entre outras [27].

Segundo Duda et al. [14], de modo geral o objetivo de um sistema de reconhecimento de padrões é atribuir classes a um conjunto de dados entrados como parâmetros. No entanto, para o sistema ser capaz de decidir qual é a classe correta de um certo dado, deve contar com um conjunto de *regras* inferidas a partir de um conjunto inicial de amostras representantes das diferentes classes, das quais conhecemos a verdadeira classe. Este conjunto é chamado de *conjunto de treinamento*. O conjunto inicial de entrada S é um conjunto de padrões representados (cada um deles) por um *vetor de características*. Esses dados de entrada são usados para aprender a sua distribuição no espaço de características, tentando estabelecer as regras de divisão das classes presentes nele. O espaço de características é um subespaço de \mathbb{R}^N , no qual cada amostra $s \in S$ é representada como um ponto. Finalmente, para classificar uma amostra nova devemos plotá-la no espaço de características e, usando as regras de divisão, decidir a qual classe pertence.

Portanto, o reconhecimento de padrões pode ser formulado como o seguinte problema: dado um conjunto S de n amostras e um conjunto C de c classes conhecidas, associar a cada amostra $s \in S$ a classe $c_s \in C$ à qual s pertence.

Muitas vezes, os dados de entrada para o sistema são um conjunto de imagens ao invés dos vetores de características, sendo necessária a aplicação de técnicas de *extração de características* para a obtenção do vetor de características. Estas características devem ser extraídas de tal modo que permitam diferenciar as diversas classes. Normalmente, é feito também um processo de *segmentação* do objeto de interesse como fase de pré-

processamento. Posteriormente, muitas das abordagens para reconhecimento de padrões tentam encaixar o objeto segmentado a um *modelo*, que não é outra coisa que um conjunto de características utilizadas para descrever o objeto e que são comuns a outros objetos semelhantes.

O passo seguinte é usar as características de cada objeto para tentar separar as classes no espaço de características. Então, o nosso problema agora é encontrar as regras para dividir o espaço, as quais podem ser representadas como um ou mais hiperplanos, um polinômio, fronteiras entre regiões, relações de conexidade em grafos, etc., chamadas de modo geral de *fronteiras de decisão*. No entanto, neste ponto é muito importante considerar que nosso objetivo final é classificar corretamente *novas amostras* e, portanto, não é necessário projetar um classificador que tenha 100% de sucesso com as amostras de treinamento, porque poderíamos ter um conjunto de regras muito complexas (se as classes se encontram misturadas) e, principalmente, porque podemos perder o poder de *generalização* do classificador sobre amostras novas. Este problema fica mais difícil se aumentamos as amostras de treinamento, pois teríamos fronteiras de decisão cada vez mais complexas. Então, é necessário manter um equilíbrio entre o número de amostras a serem usadas e a complexidade do classificador projetado.

Uma parte fundamental de todo sistema de classificação de padrões é medir o sucesso do classificador determinando sua *acurácia* na classificação das amostras novas. Esta acurácia pode ser medida de diferentes maneiras, sendo que na literatura as medidas mais utilizadas são: *verdadeiros positivos*, *verdadeiros negativos*, *falsos positivos* e *falsos negativos* (veja Seções 4.4.1 e 5.4.4). Ademais, também podemos considerar combinações dessas para obter medidas mais elaboradas que oferecem melhor informação sobre o desempenho do classificador.

Conseqüentemente, considerando os conceitos apresentados, o projeto de um sistema de classificação de padrões pode ser resumido em quatro etapas principais: geração das características, seleção das características, projeto do classificador e avaliação do classificador. As quatro etapas são apresentadas no esquema da Figura 1.1, adaptado da proposta de Theodoris e Koutroumbas [34]. Como a figura mostra, as etapas seguem uma seqüência, mas em qualquer momento podemos voltar a alguma das etapas anteriores para aprimorá-la e começar novamente a partir dessa etapa, voltando em seguida à ordem seqüencial do processo.

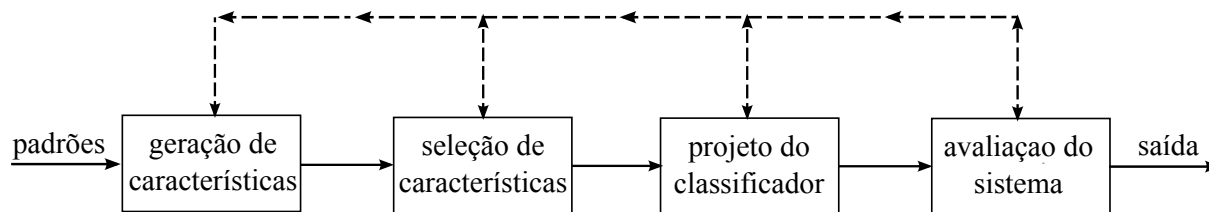


Figura 1.1: Etapas envolvidas no projeto de um sistema de classificação de padrões. (Adaptado de [34]).

Em relação à complexidade do classificador, também devemos considerar o número de características a serem usadas para representar cada padrão no espaço de características, ou seja, a dimensão desse espaço. O mais natural seria pensar que, considerando um número maior de características, vamos ter maior sucesso na classificação porque cada objeto vai estar melhor representado, porém, isso nem sempre é verdade. Algumas características podem ser redundantes, i.é., não representam nenhum benefício na classificação. Esse seria o caso quando uma característica tem perfeita correlação com outra característica. Por outro lado, podem existir características muito difíceis ou muito custosas para serem computadas, sendo que, neste caso, a melhora da acurácia na classificação pode não compensar o tempo extra de pré-processamento.

Também é importante considerar que, dependendo do sistema, as decisões feitas pelo classificador podem ter associados valores econômicos. Em outras palavras, é possível que seja extremamente importante classificar corretamente alguma das classes, sendo inaceitável a ocorrência de erros na identificação da mesma. Muitos sistemas industriais reais têm esse tipo de restrições, podendo representar uma grande perda econômica no caso de uma decisão errada.

Considerando os conceitos apresentados e os apontados por Duda et al. [14], podemos destacar entre os principais desafios para o reconhecimento de padrões: extração de características, presença de ruído, super-treinamento, seleção de um modelo adequado para representação do objeto, conhecimento *a priori* do problema, falta de algumas das características, segmentação do objeto (no caso de imagens), invariância nos extratores de características, custos associados, complexidade computacional.

1.1 Motivação do Trabalho

O presente trabalho teve como objetivo o estudo de técnicas de reconhecimento de padrões, visando à melhoria de técnicas existentes que permitam atender as demandas atuais na área. Portanto, as motivações para este trabalho foram:

- A grande variedade de aplicações que exigem a utilização de técnicas de reconhecimento de padrões.
- A quantidade crescente de grandes bases de dados, que exigem o projeto de técnicas cada vez mais eficientes.
- A existência de desafios na área de Aprendizado de Máquina para gerar classificadores eficazes e eficientes para a resolução dos problemas postos atualmente.

1.2 Problemas de interesse

Neste trabalho de mestrado, foram estudados dois importantes problemas da área de Aprendizado de Máquina, os quais são relevantes porque permitem o aprimoramento do conjunto de treinamento. Tal aprimoramento é possível através da identificação dos elementos que não devem formar parte deste conjunto e através da escolha das melhores amostras para ele.

1.2.1 Detecção de Outliers

A importância deste problema pode ser identificada a partir de dois pontos de vista. Primeiramente, um outlier é definido como um elemento dentro de uma base de dados que tem características diferentes do restante. Então, os outliers são elementos que deterioram a qualidade do conjunto de dados e, portanto, podem piorar a eficácia do classificador. Por outro lado, dependendo do domínio da aplicação, os outliers podem ser eventos raros dentro do conjunto de dados que representam eventos importantes e, justamente, serem os elementos que o especialista está procurando.

Os métodos de Detecção de Outliers presentes na literatura são normalmente baseados em medidas estatísticas, aplicadas sobre uma vizinhança de cada amostra de treinamento. O problema com esta abordagem é que o cálculo da vizinhança pode ser muito custoso, dependendo do tamanho do conjunto de dados e da complexidade da função distância empregada.

1.2.2 Construção de um Conjunto de Treinamento adequado

Este problema é de grande importância na área de aprendizado de máquina porque este conjunto é o que representa o conjunto de dados e sua escolha pode afetar a eficácia e a eficiência do classificador. A eficácia é influenciada diretamente pelo conteúdo deste conjunto, pois deve representar adequadamente o conjunto de dados; e, por outro lado, o seu tamanho impactará na eficiência do sistema.

Com respeito a este problema (eleição do conjunto), as amostras são normalmente escolhidas de forma aleatória, sem considerar sua representatividade ao problema, e o número de amostras de treinamento é arbitrariamente fixado. Tal escolha das amostras de treinamento pode afetar seriamente o processo de classificação, pois as amostras de treinamento não refletem o que realmente está presente no conjunto. Em relação ao tamanho, este parâmetro é muito difícil de se determinar e simplesmente escolher um tamanho fixo arbitrário não é uma boa estratégia a se seguir.

1.3 Proposta e Contribuições Principais

Para solucionar os problemas identificados, propomos três técnicas, concebidas como algoritmos de aprendizado supervisionado. Estes algoritmos foram desenvolvidos dentro do tópico de Classificação por Florestas de Caminhos Ótimos, que é uma metodologia que tem sido explorada com sucesso na literatura recentemente.

1.3.1 Detecção de Outliers

Dois dos algoritmos de aprendizado atacam o problema de detecção de outliers em conjuntos de dados rotulados. No primeiro algoritmo, consegue-se uma melhoria em eficácia em relação ao classificador OPF tradicional, com um pequeno incremento no tempo de processamento. No segundo, propõe-se um aprimoramento ao primeiro algoritmo, para atingir melhores taxas de detecção de outliers.

A técnica de detecção de outliers (aprimorada no segundo algoritmo) estima a medida estatística que permite definir os outliers sem computar explicitamente as vizinhanças. Esta característica permite que a técnica apresente grande eficiência computacional.

Os dois algoritmos permitiram a elaboração de dois artigos, que serão apresentados nos Capítulos 4 e 5, respectivamente.

1.3.2 Construção de um Conjunto de Treinamento adequado

O terceiro algoritmo é uma abordagem para a construção de conjuntos de treinamento (para classificação supervisionada) com tamanho reduzido e contendo as amostras mais representativas do conjunto.

Esta técnica elimina ainda a necessidade de se especificar *a priori* o tamanho do conjunto de treinamento, e permite escolher as amostras mais representativas. A eficácia do algoritmo foi confirmada pela melhoria na acurácia em relação ao classificador OPF tradicional, representando um pequeno incremento no tempo de processamento. Portanto, o algoritmo proposto se destaca pela sua eficiência e eficácia.

O algoritmo descrito permitiu a elaboração de um artigo que será apresentado no Capítulo 6.

1.4 Organização do Trabalho

Este primeiro capítulo apresentou os conceitos fundamentais de reconhecimento de padrões que são necessários para o entendimento do trabalho que foi desenvolvido. Ademais, foi apresentada a motivação do trabalho, os problemas de interesse e as contribuições.

O Capítulo 2 apresenta outros conceitos importantes para o projeto de sistemas de reconhecimento de padrões, como as etapas envolvidas no processo, a separabilidade das classes e as diferentes abordagens existentes para classificação de padrões.

No Capítulo 3 é feita uma revisão de algumas das técnicas de classificação supervisionada mais relevantes da literatura. Estas técnicas são descritas apenas com o intuito de conhecer basicamente o seu funcionamento.

No Capítulo 4 é apresentado o artigo que contém o primeiro algoritmo para Detecção de Outliers.

No Capítulo 5 é apresentado o segundo artigo sobre Detecção de Outliers, que propõe uma melhoria ao primeiro artigo.

No Capítulo 6 é apresentado o artigo que contém o algoritmo para a Construção de um Conjunto de Treinamento adequado para classificação supervisionada.

O Capítulo 7 expõe as propostas para futuras linhas de pesquisa a partir dos algoritmos apresentados nesta dissertação.

Finalmente, no Capítulo 8 são discutidas as conclusões deste trabalho.

Capítulo 2

Conceitos sobre Reconhecimento de Padrões

Como foi explicado no Capítulo 1, as principais etapas no projeto de um sistema de classificação de padrões são: geração das características, seleção das características, projeto do classificador e avaliação do classificador. Neste capítulo, serão apresentados alguns conceitos importantes para o projeto do classificador, como as etapas que o compõem, o grau de separabilidade que pode existir entre as classes que definem o problema e as abordagens que existem para resolver o problema de classificação de padrões, de acordo com a informação *a priori* disponível.

2.1 Etapas no Projeto do Classificador

Normalmente, a tarefa de reconhecimento de padrões é dividida em duas etapas, *treinamento* e *classificação*. O treinamento é o processo pelo qual o classificador aprende a distribuição dos dados através de um subconjunto das amostras disponíveis, para posteriormente ser utilizado na rotulação de novas amostras na etapa de classificação. O treinamento, geralmente, é realizado uma única vez e a classificação é feita sempre que necessária. Entretanto, existem aplicações que requerem um novo treinamento do classificador a cada instância do problema, principalmente aquelas que fazem uso da imagem do objeto a ser identificado. Em tais situações, variações comuns ao processo de aquisição de imagens fazem com que cada uma delas defina um novo problema de reconhecimento de padrões.

Alguns sistemas de reconhecimento de padrões também incluem mais uma etapa no processamento, *aprendizagem (pseudo-teste)*, a qual é utilizada para melhorar o desempenho do classificador. Para citar alguns exemplos, Papa et al. [29] propuseram uma abordagem que aumenta a eficácia de um classificador identificando as amostras mais relevantes do conjunto de treinamento a partir de um conjunto de avaliação. Ocorre que às vezes o conjunto de treinamento é muito grande, comprometendo o tempo de resposta do sistema. Papa e Falcão [28] também propuseram um algoritmo para redução do conjunto de treinamento com base em um algoritmo de aprendizagem, sem afetar significativamente a taxa de acerto no conjunto de teste.

Para realizar as três etapas mencionadas, o conjunto de dados é dividido em três subconjuntos: treinamento, avaliação e teste, os quais serão utilizados ao longo de todo o processo.

2.2 Separabilidade de Classes

Um outro aspecto importante a ser considerado em problemas de classificação é a disposição das amostras no espaço de características [29], a qual define o nível de separabilidade das mesmas, que pode ser dividido em: *linearmente separável*, *linearmente separável por partes*, *não linearmente separável* e *não separável*. A Figura 2.1 ilustra esses quatro casos.

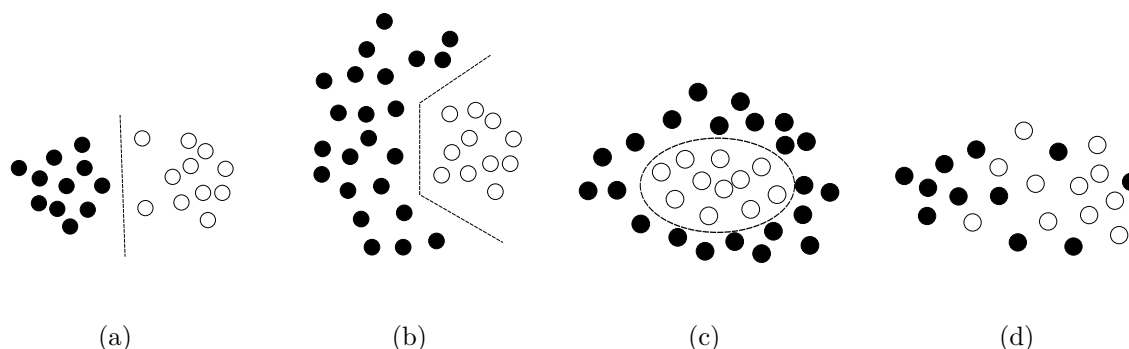


Figura 2.1: Casos de separabilidade entre duas classes representadas no espaço 2D: (a) linearmente separável, (b) linearmente separável por partes, (c) não linearmente separável e (d) não separável.

Na Figura 2.1a as amostras de diferentes classes podem ser dicotomizadas por uma reta (2D) ou um hiperplano (maior dimensão), sendo que, no caso linearmente separável por partes (Figura 2.1b), os dados podem ser separados por uma cadeia de segmentos de reta ou hiperplanos. Já no caso não linearmente separável (Figura 2.1c), a separação pode ser feita utilizando uma outra forma geométrica. No caso mais complexo, ou seja, quando as amostras não são separáveis, o desempenho dos classificadores fica prejudicado e novas técnicas são desenvolvidas com o intuito de abordar tais situações.

2.3 Abordagens para Classificação de Padrões

O problema de reconhecimento de padrões pode ser dividido em três tipos, de acordo com a quantidade de informação *a priori* disponível das amostras do conjunto de treinamento: abordagens não supervisionadas, semi-supervisionadas e supervisionadas [20].

2.3.1 Classificação Não-Supervisionada

Na classificação *não supervisionada*, também chamada de *clustering*, não são conhecidas as classes das amostras. O objetivo é formar agrupamentos de amostras (*clusters*), tais

que elementos pertencentes a um mesmo agrupamento são mais *semelhantes* entre si do que a elementos que pertençam a outros grupos [37, 22, 30]. 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. 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 [14]. Ademais, um dos maiores problemas nesta abordagem é como definir a *similaridade* entre duas amostras, assim como escolher uma medida adequada para tal similaridade.

As técnicas para classificação não-supervisionada são baseadas essencialmente em duas abordagens [24]: clustering hierárquico e clustering particional, das quais derivam a maioria das técnicas propostas na literatura. Na abordagem hierárquica, os grupos são formados gradualmente até chegar a uma partição ótima. Já na abordagem particional, o objetivo é fazer uma única partição do espaço de características. Por outro lado, também existe um importante grupo de técnicas baseadas em grafos, entre as quais podemos destacar a proposta de Zahn [38], a qual utiliza uma Árvore Geradora Mínima (*Minimum-Spanning Tree - MST*), o Grafo de Vizinhança Relativo (*Relative Neighborhood Graph - RNG*) [23], o Grafo de Gabriel (*Gabriel Graph - GG*) [23], os Grafos Limiares (*Threshold Graphs*) [21], as técnicas baseadas em Corte em Grafos (*Graph Cut*) [32, 36] e o Agrupamento por Floresta de Caminhos Ótimos (*Optimum-Path Forest Clustering - OPF Clustering*) [30, 6].

2.3.2 Classificação Semi-Supervisionada

Na classificação *semi-supervisionada*, a idéia é trabalhar com parte das amostras do conjunto de treinamento rotuladas e outra parte não. Na literatura encontram-se algumas técnicas para classificação semi-supervisionada, que são extensões de técnicas já conhecidas, tais como as Máquinas de Vetores de Suporte Transdutivas (*Transductive Support Vector Machines - TSVM*) [25], o método do Co-Treinamento [2], uma extensão do conhecido algoritmo *k*-médias. No entanto, existem vários trabalhos que demonstram que as técnicas semi-supervisionadas apresentam limitações que podem degradar muito seu desempenho. Zhu [40] demonstrou que achar uma solução exata para TSVM é um problema NP-difícil. Ghani [18] e Zhang e Oles [39] mostraram que usar dados não rotulados pode causar uma degradação de desempenho no TSVM e no Co-Treinamento, respectivamente.

2.3.3 Classificação Supervisionada

Finalmente, na classificação *supervisionada* é conhecido o rótulo de cada amostra do conjunto de treinamento. Aqui, o conjunto de amostras Z é dividido nos conjuntos Z_1, Z_2 e Z_3 , sendo que Z_1 e Z_2 são usados para projetar o classificador, e o objetivo é classificar corretamente as amostras de Z_3 . Neste caso, em aplicações de segmentação interativa, por exemplo, é fundamental um avaliador externo para a qualidade do resultado final do classificador, pois os conjuntos Z_1 e Z_2 são os marcadores feitos pelo usuário e desconhecem-se os rótulos de Z_3 .

Existem muitas técnicas de classificação supervisionada propostas na literatura. Entre as mais populares podemos citar o Classificador Bayesiano (*Bayesian Classifier - BC*) [24,

14], as Redes Neurais Artificiais (*Artificial Neural Networks - ANN*) [19], as Máquinas de Vetores de Suporte (*Support Vector Machines - SVM*) [10, 35], os k Vizinhos mais Próximos (*k Nearest Neighbors - k-NN*) [17] e a Floresta de Caminhos Ótimos (*Optimum-Path Forest - OPF*) [27, 29].

Neste trabalho é estudado o problema de *classificação supervisionada*. A Figura 2.2 ilustra o funcionamento proposto para o classificador. Primeiro é usado um conjunto Z_1 para projetar o classificador, seguido de um processo de classificação das amostras de Z_2 utilizando o classificador projetado. Tendo o resultado dessa classificação, são identificadas as amostras erroneamente classificadas em Z_2 e são trocadas por amostras aleatórias de Z_1 , processo que é chamado de aprendizagem, que dá como resultado um classificador aprimorado. Em seguida, utiliza-se este classificador aprimorado para começar novamente o treinamento, repetindo-se esse processo um certo número de vezes até se obter o classificador final. Por último, utiliza-se o classificador resultante para executar a fase de teste com as amostras do conjunto Z_3 , o que dá o resultado último do sistema.

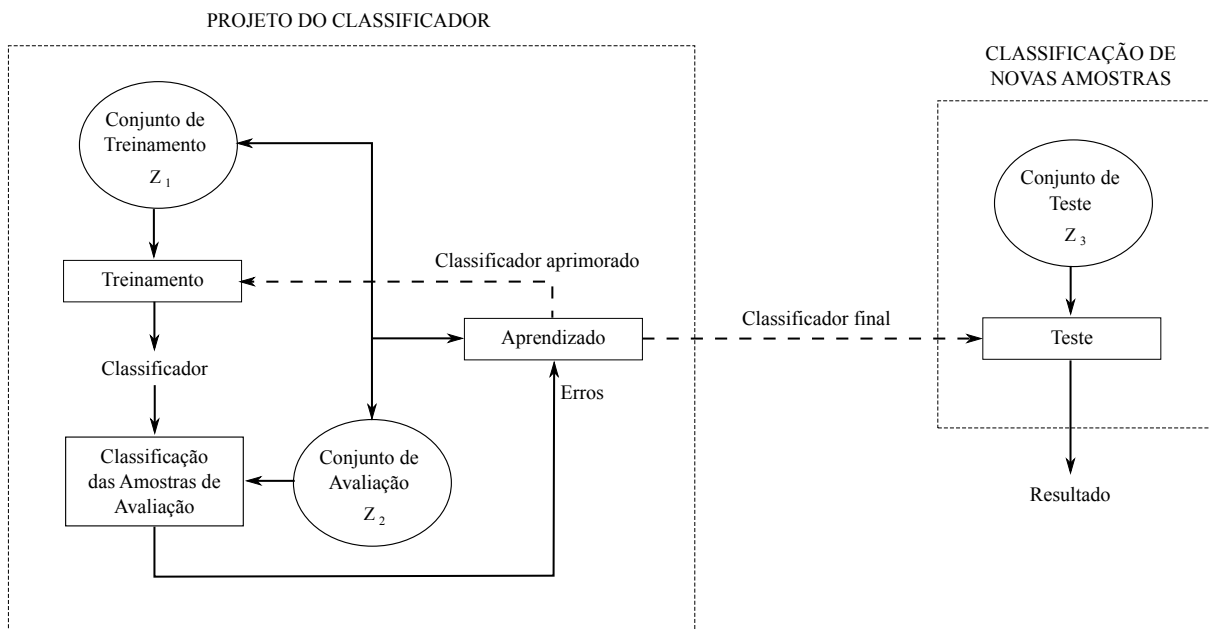


Figura 2.2: Processo seguido na classificação supervisionada, incluindo treinamento, aprendizagem e classificação de novas amostras (teste).

Capítulo 3

Revisão de Classificadores

Nesta seção, faremos uma revisão das principais técnicas de classificação supervisionada presentes na literatura.

Como foi mencionado na Seção 2.3, existem muitas técnicas de classificação supervisionada. No entanto, neste capítulo faremos uma revisão das mais importantes e mais utilizadas na literatura. Duda et al. [14] e Theodoris e Koutroumbas [34] propuseram diversas categorias para dividir estas técnicas. Entre as principais, podemos destacar as seguintes: técnicas baseadas em probabilidade (Classificador Bayesiano), técnicas não-paramétricas (k vizinhos mais próximos), discriminantes lineares (Máquinas de Vetores de Suporte) e discriminantes não-lineares (Redes Neurais Artificiais). Ademais, apresentaremos também o classificador baseado em Floresta de Caminhos Ótimos, que é uma técnica baseada em grafos proposta recentemente e que se mostrou muito eficaz e eficiente.

Nas seções seguintes, apresentaremos os principais conceitos de cada uma das técnicas escolhidas para termos um conhecimento geral das técnicas mais populares de classificação de padrões. Nas últimas seções, enfatizaremos duas técnicas que têm mais relevância para este trabalho: o Classificador por Floresta de Caminhos Ótimos e as Máquinas de Vetores de Suporte. O primeiro foi usado como base para os algoritmos de aprendizado propostos neste trabalho e o segundo será usado em trabalhos futuros.

3.1 Classificador Bayesiano

O classificador bayesiano é um classificador estatístico que usa o Teorema de Bayes [1] para descobrir a classe de um conjunto de amostras, usando informação estatística das amostras de treinamento, como as probabilidades *a priori*, *a posteriori* e as *funções de densidade de probabilidade*. Basicamente, a ideia é computar, para cada amostra, o erro desta pertencer a cada uma das classes, sendo que uma amostra é classificada na classe com menor erro e é por isso que ele também é conhecido como o *Classificador Estatístico Ótimo* [34].

3.1.1 Definições

Dado um problema com M classes $\omega_1, \dots, \omega_M$ e uma amostra representada pelo vetor de características x , definimos as probabilidades condicionais dela pertencer a cada uma das

classes como: $P(\omega_i|x)$, $i = 1, \dots, M$, chamadas de probabilidades *a posteriori*. Cada uma delas define a probabilidade da amostra pertencer a cada classe ω_i dado que o vetor tem o valor x .

Para computar a probabilidade a posteriori precisamos de outras duas probabilidades: a probabilidade *a priori* $P(\omega_i)$ de ocorrência de cada classe i e a função de densidade de probabilidade (*Probability Density Function - PDF*) $p(x|\omega_i)$, $i = 1, 2, \dots, M$, que descreve a distribuição dos vetores de características em cada uma das classes.

A probabilidade a priori pode ser calculada usando as amostras de treinamento. Seja N o número total de amostras e N_i , $i = 1, \dots, M$ o número de amostras da classe ω_i , definimos $P(\omega_i) = N_i/N$.

A PDF $p(x|\omega_i)$, também conhecida como a *função de verosimilhança* de ω_i em relação a x , também pode ser estimada a partir das amostras de treinamento, ou podemos assumir que tem uma forma conhecida (e.g., uma função gaussiana). Existem muitas abordagens para estimar a PDF.

Finalmente, podemos definir as probabilidades condicionais usando o Teorema de Bayes:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \quad (3.1.1)$$

onde $p(x)$ é a PDF de x , definida como:

$$p(x) = \sum_{i=1}^M p(x|\omega_i)P(\omega_i) \quad (3.1.2)$$

Então, usando as probabilidades, definimos que a classe de x será a classe ω_i se:

$$P(\omega_i|x) \geq P(\omega_j|x), \forall j \neq i \quad (3.1.3)$$

Naturalmente, podemos supor que todas as classes têm a mesma probabilidade de serem escolhidas (i.é., $P(\omega_i) = P(\omega_j)$, $\forall i \neq j$). Então, re-escrevendo e juntando as Equações 3.1.1 e 3.1.4 podemos deduzir que a decisão depende unicamente do valor da PDF. Então, considerando as PDFs da Figura 3.1 (duas classes), podemos estabelecer a linha que passa por x_0 como um threshold aceitável de divisão das classes. No entanto, sabemos que seria impossível evitar a ocorrência de erros nessa classificação pela superposição das PDFs, sendo que o erro pode ser medido como:

$$P_e = \int_{-\infty}^{x_0} p(x|\omega_2)dx + \int_{x_0}^{+\infty} p(x|\omega_1)dx \quad (3.1.4)$$

Logo, o objetivo do Classificador é minimizar esse erro, sendo que tal restrição é atendida na Equação 3.1.4 (mais detalhes em Theodoris e Koutroumbas [34]).

Na Figura 3.2, pode-se observar que realmente as probabilidades condicionais são um melhor critério de classificação (Equação 3.1.4) do que as PDFs.

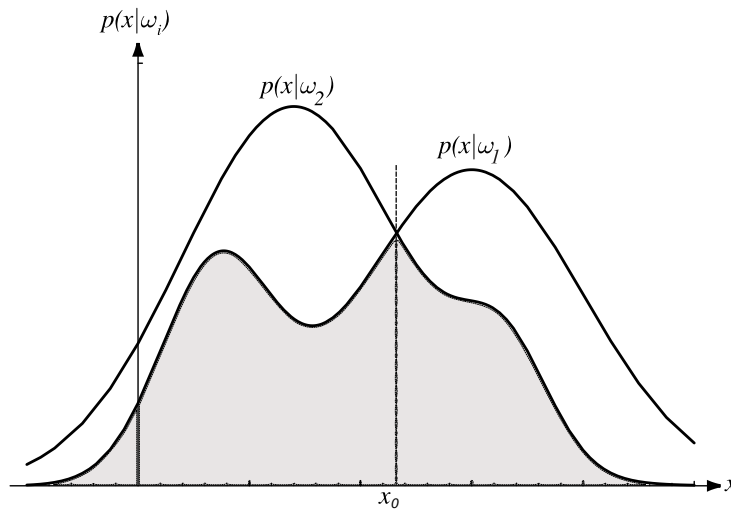


Figura 3.1: Exemplo de Funções de Densidade de Probabilidade considerando as classes ω_1 e ω_2 . O erro na classificação é representado pela região em cinza.

3.1.2 Estimativas de Funções de Densidade de Probabilidade

Como foi dito, frequentemente a PDF não é conhecida, mas pode ser estimada a partir dos dados de entrada. Estimar a PDF é um processo fundamental em estatística, pois uma PDF descreve completamente o comportamento de uma variável aleatória.

De modo geral, estimativas de densidade podem ser classificadas como paramétricas e não-paramétricas. O propósito da abordagem não-paramétrica é encontrar uma PDF desconhecida que tenha máxima verossimilhança com o conjunto de dados analisados. A alternativa paramétrica assume que os dados seguem um modelo específico de densidade. Essa alternativa pode gerar resultados não muito bons quando o conhecimento a priori não se adequa muito bem aos dados. Por outro lado, quando se conhece uma função que caracteriza bem o conjunto de dados, o método pode ser muito mais rápido que o não-paramétrico.

Entre as principais técnicas para estimativa de PDF podemos destacar as apontadas por Theodoris e Koutroumbas [34]: estimativa por máxima verossimilhança, estimativa por probabilidade *a posteriori*, inferência bayesiana, estimativa por máxima entropia, maximização da esperança (mistura de PDFs), janelas de Parzen, histogramas, k vizinhos mais próximos, núcleo variável.

Até agora foram apresentados os conceitos básicos do Classificador Bayesiano, unicamente para conhecer a lógica atrás do seu funcionamento. Existem muitos outros aspectos que foram estudados ao longo dos anos na literatura, mas que não são relevantes ao conteúdo deste trabalho. Aspectos como a Minimização do Risco na classificação, diferentes Superfícies de Decisão, estimativas de Funções de Densidade de Probabilidade, etc, podem ser encontradas em Duda et al. [14] e Theodoris e Koutroumbas [34].

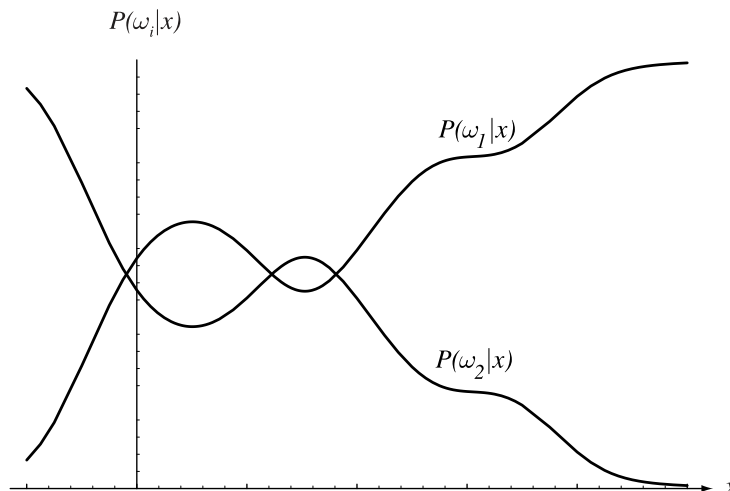


Figura 3.2: Exemplo de Probabilidades Condicionais para as PDFs da Figura 3.1, considerando $P(\omega_1) = 2/3$ e $P(\omega_2) = 1/3$. Para cada x , $P(\omega_1) + P(\omega_2) = 1$.

3.2 k Vizinhos mais Próximos

O classificador k -NN é caracterizado pela sua simplicidade conceitual e o bom desempenho que pode mostrar em muitos problemas de classificação. A ideia de utilizar os k vizinhos mais próximos de cada amostra foi inicialmente proposto como uma técnica não-paramétrica para o cálculo de funções de densidade de probabilidade, sendo posteriormente definido como uma regra de classificação. Segundo Theodoris e Koutroumbas [34], este é um classificador não-linear sub-ótimo, pois representa uma aproximação da PDF real dos dados, computada em muito menos tempo.

Dado um conjunto de N vetores de características, um conjunto de M classes, um vetor x que representa uma nova amostra a ser classificada e uma medida de distância, o algoritmo de classificação k -NN pode ser resumido nos seguintes passos

1. Identificar as k amostras mais próximas a x , independentemente das classes dessas amostras.
2. Computar o número de amostras k_i que pertencem a cada classe ω_i , $i = 1, \dots, M$.
3. Atribuir x à classe ω_i com o maior número de amostras k_i .

Para estudar alguns aspectos interessantes do classificador k -NN, vamos considerar a versão mais simples possível, i.é, $k = 1$, sendo que neste caso, o algoritmo seria reduzido a considerar que a classe de x será aquela da amostra mais próxima a ela. Então, neste caso, o classificador representa uma partição do espaço em N regiões R_i , sendo que cada uma delas é definida por:

$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\} \quad (3.2.5)$$

ou seja, R_i contém todos os pontos mais próximos a x_i do que a qualquer outro ponto no conjunto, considerando a distância d . Esta partição do espaço é conhecida como o Diagrama de Voronoi (Figura 3.3).

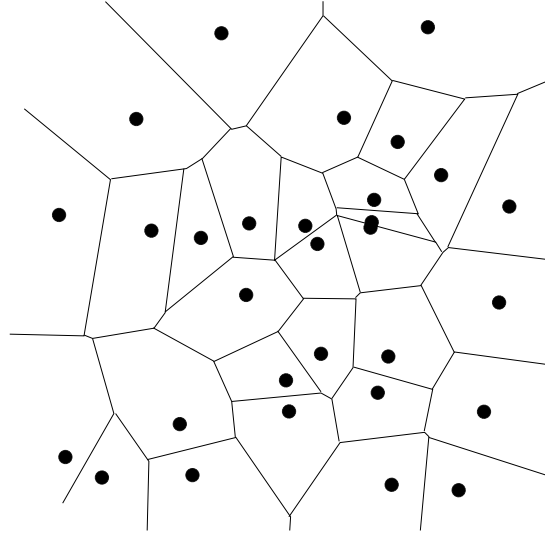


Figura 3.3: Exemplo de um Diagrama de Voronoi considerando $N = 30$ em um espaço 2D e usando distância euclidiana.

Se temos um número suficientemente grande de amostras, o comportamento do classificador pode ser bastante bom. Mais especificamente, se $N \rightarrow \infty$, o erro de classificação P_{NN} do classificador é limitado por:

$$P_B \leq P_{NN} \leq P_B \left(2 - \frac{M}{M-1} P_B \right) \leq 2P_B \quad (3.2.6)$$

onde P_B é o erro de classificação Bayesiano (i.é., erro ótimo) [14]. Em outras palavras, o erro do classificador 1-NN é no máximo duas vezes o erro ótimo.

Por outro lado, se k é suficientemente grande, o classificador k -NN é melhor que o classificador 1-NN. Para o caso de $M = 2$, por exemplo, Devroye et al. [12] estabeleceram as seguintes desigualdades:

$$P_B \leq P_{k-NN} \leq P_B + \frac{1}{\sqrt{ke}}$$

$$P_B \leq P_{k-NN} \leq P_B + \sqrt{\frac{2P_{NN}}{k}} \quad (3.2.7)$$

sendo que as duas desigualdades sugerem que quando k cresce, o comportamento do classificador k -NN tende ao ótimo. Além disso, Theodoris e Koutroumbas [34] também demonstraram esta aproximação do classificador k -NN ao classificador Bayesiano ótimo.

Contudo, um dos grandes problemas deste classificador é a complexidade computacional para o cálculo dos vizinhos mais próximos das amostras no conjunto de treinamento.

Se fôssemos usar um algoritmo de força bruta, a complexidade seria $O(kN)$, o qual pode ser inviável para muitos problemas na prática. Ademais, o problema se complica ainda mais ao trabalharmos com vetores de características de altas dimensões. Na literatura foram propostas algumas alternativas para reduzir o tempo de processamento do classificador, tais como as abordagens de Broder [3], Djouadi e Bouktache [13], Nene e Nayar [26] e Dasarathy [11].

3.3 Redes Neurais Artificiais

O projeto de classificadores baseados em Redes Neurais [19] vem de uma metodologia inspirada no funcionamento do cérebro humano e é amplamente estudado na literatura, tanto pelas suas propriedades teóricas quanto pelas suas aplicações práticas. Basicamente, trabalha-se com um conjunto de neurônios (unidades de processamento) interligados para formar uma rede de nós, organizados em diferentes camadas, como pode-se observar no exemplo da Figura 3.4. O primeiro conjunto de neurônios (camada de entrada) realiza algum processamento com o conjunto de treinamento e produz um conjunto de saídas que servem como entradas para os neurônios da camada intermediária. Estes neurônios realizam um processamento similar e produzem novas saídas para a camada seguinte, repetindo-se o mesmo processo com as camadas posteriores. A última camada (camada de saída) deve conter um neurônio por cada classe do problema, os quais definem a classe de amostras desconhecidas, como veremos nas próximas seções.

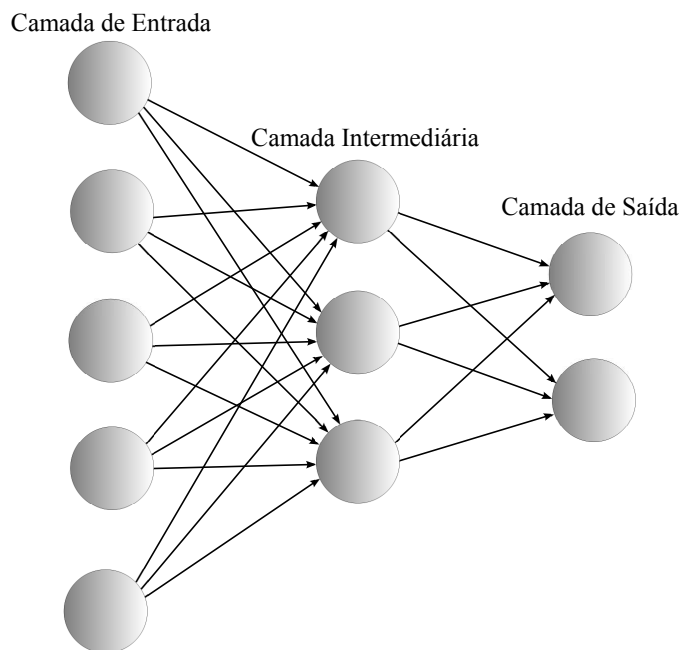


Figura 3.4: Exemplo de uma Rede Neural Artificial com três camadas e duas saídas.

Existem diversas propostas relacionadas à arquitetura da rede, sendo que essa arquitetura é que define o modo em que os neurônios são interligados e se existirá mais de

uma camada. Ademais, cada uma das interligações entre os neurônios possui um certo peso que é usado na computação do neurônio. Comumente, são utilizados três tipos de arquiteturas de redes neurais:

1. As redes *feedforward*, que unicamente transformam sinais de entrada em sinais de saída (usados na seguinte camada da rede).
2. As redes *feedback*, que igualmente transformam os sinais de entrada para serem usados na camada seguinte, mas também podem utilizar os resultados de uma camada para alimentar neurônios de camadas anteriores, realizando iterações no processo até atingir um estado final.
3. Os *mapas de auto-organização*, que permitem interações entre neurônios vizinhos na mesma camada, a fim de transformarem-se em detectores especializados de diferentes padrões.

3.3.1 Perceptrons

Um conceito fundamental que revolucionou a pesquisa em redes neurais nas décadas de 50 e 60 é a invenção das chamadas máquinas que aprendem, os *perceptrons* [19]. Essencialmente, um perceptron aprende uma função de decisão linear que dicotomiza dois conjuntos de treinamento, para o qual utiliza o conjunto de pesos de entrada que chegam a ele através das interligações com outros perceptrons (neurônios).

A saída $d(x)$ de um perceptron x é definida por uma *função de ativação* $\phi(\xi)$ que tem como parâmetro a soma ponderada ξ de sua entrada, ou seja,

$$d(x) = \phi(\xi)$$

$$\xi = \sum_{i=1}^n w_i x_i + w_{n+1} \quad (3.3.8)$$

onde os coeficientes $w_i, i = 1, \dots, n + 1$ são os pesos das conexões. A soma ponderada ξ define um hiperplano no espaço n -dimensional.

A função de ativação ϕ pode ter diversas formas, sendo as mais conhecidas: função limiar, sigmoidal, identidade, etc.

- Função limiar:

$$\phi(\xi) = \begin{cases} 1 & \text{se } \xi \leq 1 \\ 0 & \text{caso contrário} \end{cases}$$

- Função sigmoidal:

$$\phi(\xi) = \frac{1}{1 + \exp(-\xi)}$$

- Função identidade:

$$\phi(\xi) = \xi$$

Utilizando o hiperplano definido por ξ , podemos resolver problemas de classificação que envolvem duas classes linearmente separáveis. No entanto, quando tentamos resolver problemas com mais de duas classes (linearmente separáveis ou não), esta técnica não é eficiente. Para resolver esta inconveniência, foi proposto o algoritmo Perceptron Multicamadas, conhecido como ANN-MLP (*Artificial Neural Networks - Multilayer Perceptron*) [14, 19].

Nas redes ANN-MLP, temos um conjunto de camadas de neurônios, sendo que o conjunto de saídas de uma camada alimenta as entradas dos neurônios da camada seguinte. Para cada neurônio, sua entrada corresponde à soma ponderada das saídas da camada anterior.

O número de neurônios da primeira camada corresponde à dimensionalidade dos vetores de características dos dados de entrada; e o número de neurônios da camada de saída corresponde ao número de classes presentes no problema. Logo, quando um novo padrão é apresentado à rede, definimos que sua classe é w_m caso a m -ésima saída da rede possua o maior valor de todas as saídas.

3.3.2 Treinamento por Retropropagação

O algoritmo ANN-MLP descrito anteriormente possui uma arquitetura *feedforward*, pois os neurônios na rede são ligados unicamente com neurônios de camadas posteriores. No entanto, utilizando uma arquitetura *feedback*, podemos melhorar o desempenho da rede utilizando nas primeiras camadas informações obtidas em camadas posteriores. Este é o chamado Treinamento por Retropropagação [4].

Basicamente, o objetivo deste algoritmo é minimizar o erro quadrático total entre as saídas desejadas e as saídas obtidas pela rede na última camada. Para isso, utilizamos um conjunto de aprendizado do qual conhecemos as classes e classificamos cada uma das amostras utilizando os pesos definidos na rede. Caso a saída obtida pela rede não esteja correta, calculamos o erro e o propagamos da camada de saída até a camada de entrada e desta até as outras, utilizando a *Regra Delta Generalizada*, a qual permite o reajuste dos pesos em toda a rede, de maneira a tentar minimizar a função de erro (veja [4] para mais detalhes).

Para a camada de saída Q , o erro quadrático total E_Q é definido por:

$$E_Q = \frac{1}{2} \sum_{q=1}^{N_Q} (r_q - O_q)^2 \quad (3.3.9)$$

onde N_Q é o número de nós na camada Q , r_q é a saída desejada e O_q é a saída obtida pela rede.

Uma vez que todas as amostras no conjunto de aprendizado tenham sido classificadas, podemos usar os pesos finais da rede para classificar novas amostras com classe desconhecida.

Como foi dito no começo da seção, as Redes Neurais são uma das técnicas de classificação mais estudadas na literatura e, como tal, foram propostos muitos outros algoritmos de treinamento e arquiteturas além dos apresentados nesta seção. No entanto, o nosso objetivo foi apenas introduzir alguns conceitos sobre redes neurais por ser uma técnica

muito conhecida. Mais detalhes sobre os algoritmos e arquiteturas apresentados e outros existentes, podem ser encontradas em Haykin [19], Bryson e Ho [4], Duda et al. [14] e Theodoris e Koutroumbas [34].

3.4 Máquinas de Vetores de Suporte

O classificador baseado em Máquinas de Vetores de Suporte é um classificador que essencialmente busca dividir um conjunto de dados através de um hiperplano escolhido por otimização de um conjunto de possíveis divisores. Porém, como nem sempre os dados estarão linearmente divididos no seu espaço de características original, eles são mapeados a espaços de maior dimensão onde se assume que eles podem ser linearmente divididos.

3.4.1 Classificação Binária

O classificador SVM foi proposto para definir problemas de classificação binária, que podem ser definidos por:

$$(x_1, y_1), \dots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}$$

onde o objetivo é atribuir uma classe já conhecida (+1 ou -1) às *entradas* x_i , formando assim as *saídas* y_i .

A distância (medida de similaridade) entre dois vetores $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$ é definida como o comprimento do vetor diferença $\mathbf{y} = \mathbf{x}' - \mathbf{x}$ entre eles, o qual é definido através do *produto interno*: $\|\mathbf{y}\| = \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle}$.

Para obter uma melhor separação entre os dados, os vetores de características são mapeados a um espaço de maior dimensão. Um tal mapeamento Φ pode ser definido para os espaços \mathcal{X} e \mathcal{H} como $\Phi : \mathcal{X} \rightarrow \mathcal{H}$, tal que, $x \rightarrow \mathbf{x} = \Phi(x)$

Então, baseando-nos nos conceitos apresentados, podemos propor um algoritmo geométrico para classificação binária [31], que é uma base para o classificador baseado em SVM. A ideia básica é atribuir uma de duas classes às amostras desconhecidas, baseado na menor distância entre a amostra e as médias das classes.

As médias \mathbf{c}_+ e \mathbf{c}_- das duas classes são definidas por:

$$\mathbf{c}_+ = \frac{1}{m_+} \sum_{\forall i | y_i = +1} \mathbf{x}_i \quad , \quad \mathbf{c}_- = \frac{1}{m_-} \sum_{\forall i | y_i = -1} \mathbf{x}_i \quad (3.4.10)$$

onde m_+ e m_- são o número de amostras de cada classe.

A classificação é feita verificando-se, para uma nova amostra \mathbf{x} , se o vetor $\mathbf{x} - \mathbf{c}$ forma um ângulo menor que $\pi/2$ com o vetor $\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-$, normal ao hiperplano de decisão. O ponto \mathbf{c} é definido como o ponto médio entre as médias das classes. Logo, a classificação é feita usando o sinal do produto interno, definindo assim, a *função de decisão* ou *classificação*, a qual induz uma fronteira de decisão, que tem a forma de um hiperplano de dimensão N , definido por:

$$y = \text{sgn}(\langle \mathbf{x} - \mathbf{c}, \mathbf{w} \rangle) = \text{sgn}(\langle \mathbf{x}, \mathbf{c}_+ \rangle - \langle \mathbf{x}, \mathbf{c}_- \rangle + b) \quad (3.4.11)$$

onde b é um deslocamento definido por:

$$b = \frac{1}{2}(\|\mathbf{c}_-\|^2 - \|\mathbf{c}_+\|^2) \quad (3.4.12)$$

Os vetores \mathbf{c}_+ , \mathbf{c}_- e \mathbf{w} são obtidos geometricamente a partir dos vetores \mathbf{x}_i , porém, não podem ser usados diretamente para calcular o produto interno pois têm que ser expressados em termos do conjunto de entrada x_1, \dots, x_m . Para expressá-los desta forma é usada a função k , chamada de *função núcleo*, que representa o produto interno do mapeamento das amostras, definido por: $k(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle$. Então, usando k e (3.4.10), a função de decisão (3.4.11) fica assim:

$$y = \text{sgn} \left(\frac{1}{m_+} \sum_{\forall i|y_i=+1} k(x, x_i) - \frac{1}{m_-} \sum_{\forall i|y_i=-1} k(x, x_i) + b \right) \quad (3.4.13)$$

e o deslocamento (3.4.12):

$$b = \frac{1}{2} \left(\frac{1}{m_-^2} \sum_{\forall (i,j)|y_i=y_j=-1} k(x_i, x_j) - \frac{1}{m_+^2} \sum_{\forall (i,j)|y_i=y_j=+1} k(x_i, x_j) \right) \quad (3.4.14)$$

Neste classificador, os somatórios de cada classe são multiplicados por um peso ($\frac{1}{m_+}$ ou $\frac{1}{m_-}$) considerando desta forma, o mesmo peso para todas as amostras da classe $+1$, assim como para a classe -1 . Porém, dependendo do problema, as amostras podem ter pesos diferentes. É por isso que neste caso temos que considerar um conjunto de pesos α_i para $i = 1, \dots, m$ para cada vetor, ficando o classificador assim:

$$y = \text{sgn} \left(\sum_{i=1}^m \alpha_i k(x, x_i) + b \right) \quad (3.4.15)$$

Existem algumas amostras que terão $\alpha_i = 0$, ou seja, elas não serão consideradas na classificação de novas amostras. No espaço de características, isto é equivalente a dizer que o vetor w , normal ao hiperplano de fronteira será representado como uma combinação linear dos padrões de treinamento (com coeficientes não uniformes). Esta representação não-uniforme significa que essas amostras não contribuem na melhora da classificação e sendo assim, a classificação vai depender somente de algumas amostras (vetores), as quais são chamadas de *Vetores de Suporte (Support Vectors)*.

3.4.2 Classificação por Hiperplano Ótimo

O classificador SVM pode ser formulado como uma extensão do classificador geométrico apresentado na seção anterior. Nesse algoritmo, tem que ser escolhido um *hiperplano ótimo de divisão* que é distinguido por atingir a máxima separação entre qualquer ponto de treinamento e ele [35]. Os pontos \mathbf{x} que estão nesse hiperplano de decisão são definidos por $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, então, a escolha é definida por:

$$\max_{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}} \left\{ \min_{\forall i=1, \dots, m} \{ \|\mathbf{x} - \mathbf{x}_i\| \mid \mathbf{x} \in \mathcal{H}, \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \} \right\} \quad (3.4.16)$$

onde \mathbf{w} é o vetor normal ao hiperplano.

Logo, esse problema pode ser reformulado utilizando apenas o comprimento do vetor normal. Segundo a definição dos pontos que estão no hiperplano de separação, os pontos que pertencem a cada classe são definidos por:

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq +1 \quad (\text{quando } y_i = +1) \quad (3.4.17)$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \quad (\text{quando } y_i = -1). \quad (3.4.18)$$

Os pontos que estão na fronteira das classes são definidos por $H_1 : \langle \mathbf{w}, \mathbf{x}_i \rangle + b = +1$ e $H_2 : \langle \mathbf{w}, \mathbf{x}_i \rangle + b = -1$ (Figura 3.5), ou seja, são paralelos pois têm a mesma normal. Então, podemos atingir a máxima separação definida por (3.4.16), da seguinte forma:

$$\min_{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\} \quad (3.4.19)$$

que é uma *otimização restrita*, sujeita às *restrições de desigualdade*:

$$y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \geq 0, \forall i = 1, \dots, m \quad (3.4.20)$$

que representam os conjuntos de desigualdades (3.4.17) e (3.4.18) em um conjunto só.

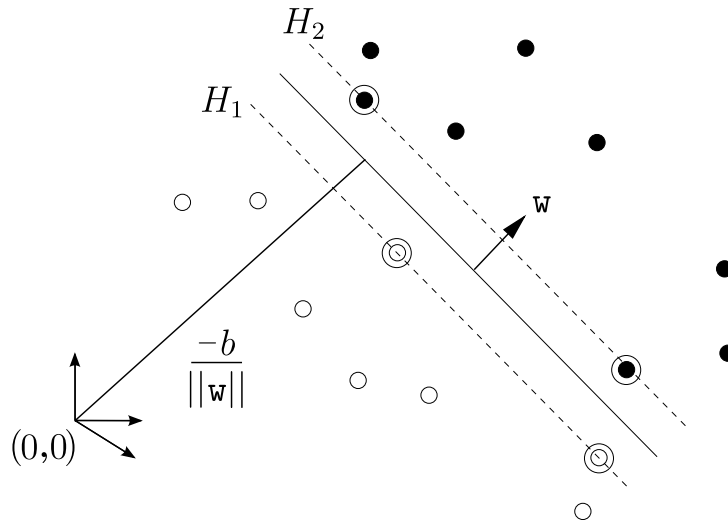


Figura 3.5: Representação do hiperplano divisor, atingindo a máxima separação com os pontos de treinamento através dos *vetores de suporte* (ressaltados com círculos), os quais definem os hiperplanos H_1 e H_2 . (Extraído de [5]).

Na prática, a otimização da Equação 3.4.19 é resolvida transformando-a em um Lagrangiano [5], que posteriormente é transformado em sua Formulação Dual equivalente,

chamada de *Wolf Dual* [16]. Porém, para este trabalho apenas apresentaremos o classificador como o problema de Otimização Restrita, definido como uma extensão do Classificador Binário (para mais detalhes, consultar [5]).

3.5 Floresta de Caminhos Ótimos

A técnica de classificação supervisionada baseada em florestas de caminhos ótimos [27] 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 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.

3.5.1 Definições

Seja um conjunto de amostras Z , o qual é dividido em três sub-conjuntos disjuntos: Z_1 que é usado para projetar o classificador (treinamento); Z_2 que é usado para o processo de aprendizagem (avaliação); e Z_3 que é usado para fazer a classificação propriamente dita (teste), tal que $Z = Z_1 \cup Z_2 \cup Z_3$.

Seja $\lambda(s)$ uma função que associa a classe correta i para $i = 1, \dots, c$ a uma amostra $s \in Z_1 \cup Z_2 \cup Z_3$, entre c possíveis classes. O conjunto de protótipos de todas as classes é representado por $S \subset Z_1$.

Para representar as amostras no grafo, é usado um algoritmo v que extrai n características de qualquer amostra $s \in Z_1 \cup Z_2 \cup Z_3$, e retorna o vetor de características $\vec{s} = v(s)$. A distância $d(s, t)$ entre duas amostras s e t é calculada como a distância entre os seus respectivos vetores de características, sendo possível usar qualquer função de distância.

Desta forma, o problema consiste em projetar um classificador que possa associar corretamente a classe $\lambda(s)$ para todas as amostras $s \in Z_3$. A OPF cria uma divisão discreta ótima do espaço de características, tal que é possível classificar qualquer amostra $s \in Z_3$ segundo essa divisão discreta. A partição é uma Floresta de Caminhos Ótimos calculada sobre Z_1 usando o algoritmo da IFT [15].

Seja (Z_1, A) o grafo completo que representa o conjunto de treinamento Z_1 , tal que todo par de amostras define um arco em A , sendo $A = Z_1 \times Z_1$. Um caminho é representado como uma sequência de amostras distintas $\pi_t^s = \langle s = t_1, t_2, \dots, t_n = t \rangle$. Esse caminho é chamado de *trivial* se $\pi_t^s = \langle t \rangle$ (i.e., $s = t$); e a *concatenação* entre um caminho $\pi_{s'}^s$ e um arco (s', t) é definida como $\pi_{s'}^s \cdot \langle s', t \rangle$. Para associar um custo a um caminho π_t^s , é usada uma função de conectividade $f(\pi_t^s)$. Um caminho π_t^s é ótimo se, para qualquer outro caminho $\pi_t^{s'}$, $f(\pi_t^s) \leq f(\pi_t^{s'})$.

Neste trabalho, foi escolhida a função de conectividade f_{max} , a qual calcula o arco de maior peso no caminho $\pi_{s'}^s \cdot \langle s', t \rangle$; sendo definida da seguinte maneira:

$$f_{max}(\langle s \rangle) = \begin{cases} 0 & \text{se } s \in S \\ +\infty & \text{caso contrário.} \end{cases}$$

$$f_{max}(\pi_{s'}^s \cdot \langle s', t \rangle) = \max\{f_{max}(\pi_{s'}^s), d(s', t)\} \quad (3.5.21)$$

O algoritmo de construção da OPF minimiza a função f_{max} para cada amostra $t \in Z_1$, de acordo com a função:

$$C(t) = \min_{\forall \pi_t^s \text{ em } \Pi(Z_1, A, t)} \{f_{max}(\pi_t^s)\} \quad (3.5.22)$$

associando para todo $t \in Z \setminus S$ um caminho ótimo π_t^* de algum $s \in S$ até t . Então, a OPF é a união de todos tais caminhos. $\Pi(Z_1, A, t)$ representa o conjunto de todos os caminhos em (Z_1, A) que terminam em t . A raiz $R(t) \in S$ do caminho π_t^* pode ser obtida seguindo os antecessores ao longo do caminho.

O classificador apresentado é uma Floresta de Caminhos Ótimos que têm como raízes os protótipos das classes, ou seja, cada amostra pertence a uma árvore de caminho ótimo que tem como raiz o seu protótipo mais fortemente conexo. Para classificar uma amostra, são calculados os caminhos ótimos dos protótipos até a amostra e é associado aquele que tem o menor peso.

3.5.2 Treinamento

Seja S o conjunto de protótipos ótimos para o classificador, isto é, o conjunto que minimiza os erros na classificação do conjunto Z_1 . O treinamento consiste em encontrar o conjunto S e construir uma OPF com raízes em S .

Para encontrar o conjunto S pode-se usar muitas heurísticas, por exemplo, uma escolha aleatória; porém, tal escolha poderia prejudicar o desempenho do classificador. Neste trabalho, S é definido calculando a árvore geradora mínima (MST) sobre o grafo completo (Z_1, A) , obtendo assim, um grafo conexo acíclico que tem como nós as amostras de Z_1 e como pesos nas arestas a distância entre amostras adjacentes. Pelo fato de a MST ser ótima em comparação com qualquer outra árvore geradora (i.é., a soma dos pesos das suas arestas é mínima) e como a função de conexidade usada é a definida na Equação 3.5.21, então existe um caminho na MST entre qualquer par de amostras que é mínimo, ou seja, a MST contém um caminho ótimo para cada amostra em Z_1 .

Logo, os protótipos são definidos como os pares de amostras conexos na MST, mas que pertencem a diferentes classes em Z_1 (fronteiras entre classes); então, as arestas que ligam esses pares são apagadas e os pares tornam-se protótipos ótimos. Segundo essa definição, uma classe pode ser representada por um conjunto de protótipos ótimos (i.é. árvores de caminhos ótimos) que podem estar separados, o qual faz com que a OPF possa resolver o problema de classes não separáveis linearmente.

Finalmente, computa-se a Floresta de Caminhos Ótimos utilizando a Equação 3.5.22 e os protótipos escolhidos.

3.5.3 Classificação

A classificação é feita calculando-se para cada amostra $t \in Z_3$ todos os possíveis caminhos até as amostras $s \in Z_1$ e depois é calculado para cada t o caminho ótimo π_t^* com raiz em S , sendo que t é classificado como sendo da classe $\lambda(R(t))$ do seu protótipo mais fortemente conexo $R(t) \in S$. O caminho é calculado incrementalmente, cujo custo é dado por:

$$C(t) = \min_{\forall s \in Z_1} \{\max\{C(s), d(s, t)\}\} \quad (3.5.23)$$

ou seja, usando $C(s)$ é calculado para cada $s \in Z_1$, o custo de $R(s)$ até s , sendo que o custo final até s será aquele que minimiza esse custo.

Seja $s^* \in Z_1$ a amostra que satisfaz a Equação 3.5.23, ou seja, $\pi_t^{R(s^*)}$ é um caminho ótimo; temos que $L(s^*) = \lambda(R(s^*))$. A classificação simplesmente é atribuir $L(s^*)$ como a classe de t (i.é., $L(t) = L(s^*)$), sendo que um erro se produz quando $L(s^*) \neq \lambda(t)$.

A Figura 3.6 apresenta um exemplo do processo inteiro descrito anteriormente, considerando duas classes distintas. O exemplo contém o Grafo inicial (Z_1, A) , a MST computada a partir desse grafo, os protótipos escolhidos da MST, a OPF resultante e a classificação de uma nova amostra utilizando a OPF computada. Os valores em cada uma das arestas $\langle s, t \rangle \subset (Z_1, A)$ nas Figuras 3.6a, 3.6b, 3.6c e 3.6e, são a distância $d(s, t)$ entre as amostras s e t . Os valores acima de cada amostra $s \in Z_1$ nas Figuras 3.6d, 3.6e e 3.6f, representam o custo $C(s)$ que foi atribuído a s quando foi computado o caminho ótimo π_s^* , usando as Equações 3.5.22 e 3.5.23, para as etapas de treinamento e classificação, respectivamente.

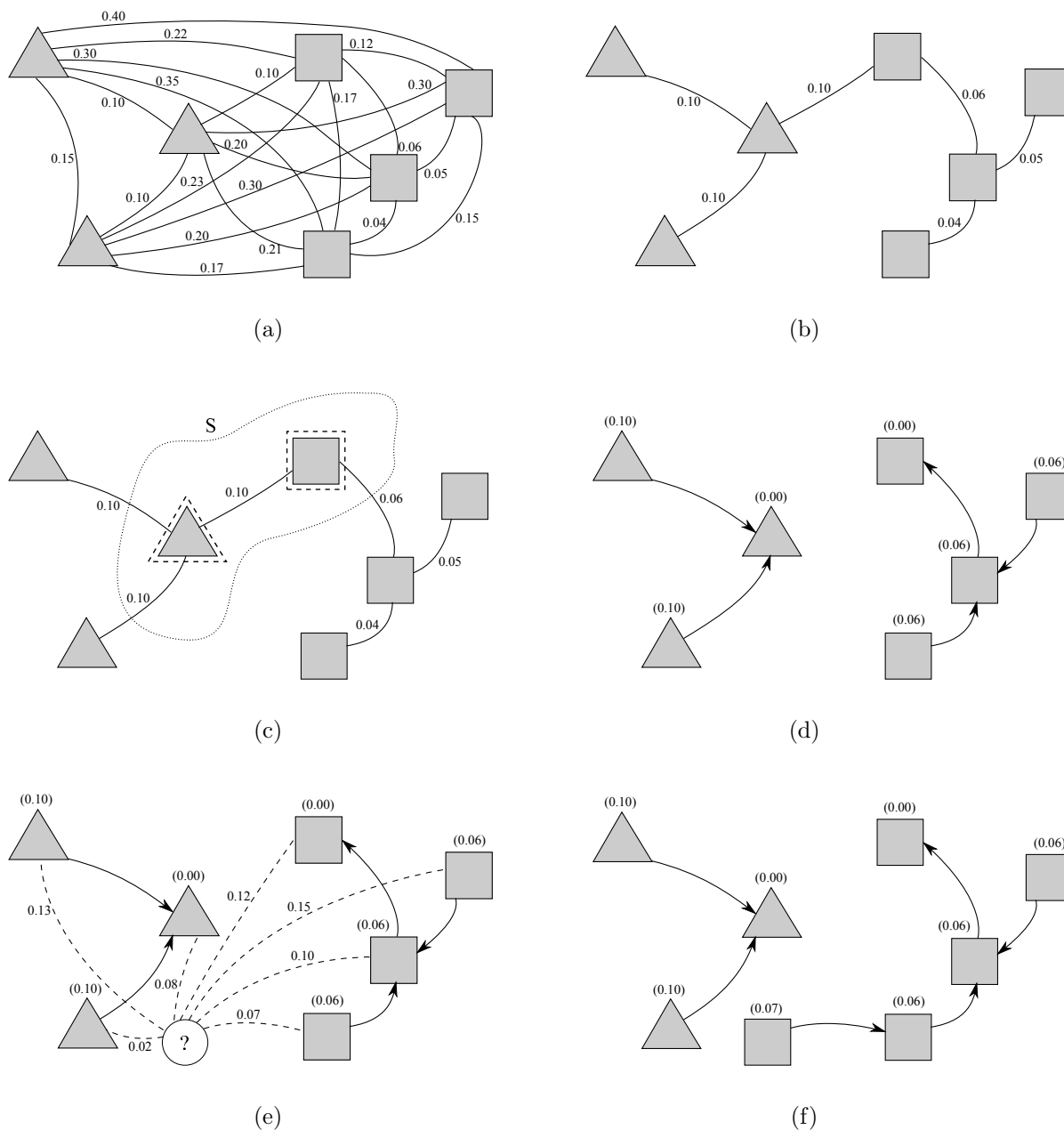


Figura 3.6: (a) Grafo completo (Z_1, A) gerado para o conjunto de treinamento Z_1 . (b) Árvore MST gerada a partir do grafo completo (Z_1, A) . (c) Escolha do conjunto de protótipos S a partir da MST. (d) Floresta de Caminhos Ótimos gerada para o grafo completo (Z_1, A) . (e) Classificação de uma nova amostra: calculam-se os caminhos até cada uma das amostras em Z_1 . (f) Escolha do caminho ótimo para a nova amostra.

Capítulo 4

Improving the Accuracy of the Optimum-Path Forest Supervised Classifier for Large Datasets

Prólogo

O artigo incluído neste capítulo, em co-autoria com Pedro J. de Rezende e Alexandre X. Falcão, ambos do Instituto de Computação da Universidade Estadual de Campinas e João Paulo Papa, do Departamento de Ciência da Computação da Universidade Estadual Paulista, campus Baurú, foi apresentado em São Paulo em novembro de 2010, durante o 15th CIARP (*Iberoamerican Congress on Pattern Recognition*) e publicado pela *Springer-Verlag* nos anais do mesmo congresso.

Este artigo apresenta um novo algoritmo de aprendizado para o Classificador baseado em Floresta de Caminhos Ótimos que busca melhoria em acurácia através da detecção de outliers no conjunto de treinamento, trocando-os por novas amostras do conjunto de avaliação. O algoritmo computa uma penalidade para cada amostra para decidir se esta é um outlier, baseada no seu comportamento como amostra classificadora (através do número de Falsos Positivos, Falsos Negativos, Verdadeiros Positivos e Verdadeiros Negativos).

O método obteve bons resultados nos conjuntos de dados usados para os testes, conseguindo melhorar a acurácia do Classificador OPF tradicional para a grande maioria dos casos, precisando apenas de um pequeno acréscimo no tempo de treinamento e conseguindo uma melhora no tempo de classificação.

Abstract

In this work, a new approach for supervised pattern recognition is presented which improves the learning algorithm of the Optimum-Path Forest classifier (OPF), centered on detection and elimination of outliers in the training set. Identification of outliers is based on a penalty computed for each sample in the training set from the corresponding number of imputable false positive and false negative classification of samples. This approach enhances the accuracy of OPF while still gaining in classification time, at the expense of a slight increase in training time.

4.1 Introduction

Pattern recognition aims at the capacity of classifying a pattern based in its inherent characteristics, represented as a feature vector (i.e., a point in a higher dimensional space) [5]. Usually, this task is divided into two phases, namely, training and classification. During the training phase the classifier learns the distribution of the data in the feature space through a subset of the dataset, inferring rules which then allow for predicting the correct classes of unknown data. Several methods also include a learning phase, based on two labeled sets with training and evaluating samples, which are usually selected at random, with the goal of improving the performance of the classifier. This phase employs the evaluating set as a pseudo-test of the quality of the training set and for its improvement.

Support Vector Machines (SVM) [9], a largely used classification method, is formulated as an optimization problem that seeks to determine the hyperplane which best splits the data. Also, given non-separable data, a prior mapping to a higher dimensional space is required, assuming that the mapped data becomes separable. SVM's main deficiency is that, depending on the size of the training set, too much computational time is needed before convergence to a solution occurs. This lack of efficiency for large datasets can render SVM infeasible in these cases. Furthermore, the assumption of class-separability in higher-dimensional space may not always hold [4].

The Optimum-Path Forest classifier (OPF) [8] is a graph-based technique which models classification problems as optimum-path searches in graphs derived from an adjacency relation between samples in a given feature space (a complete relation in the case of this paper). Class representatives (prototypes) are chosen among the training samples and used to classify the remaining samples based on lengths of paths on the graph. This method has as an advantage a very low computational training cost, since it does not have to optimize parameters. Furthermore, it can deal with non-separable data since its formulation is based on multiple prototypes that represent the various classes.

As showed in [8], OPF can attain an accuracy as high as SVM's, while keeping the training time much lower. Those results show that OPF is the best choice for several classification problems, mainly in the case of large datasets. However, a disadvantage of the original learning algorithm proposed in [8], is that it does not attempt to eliminate outliers in the training set which may cause classification errors.

The main contribution of the present work is the development of a learning algorithm for the OPF classifier focused on the detection and elimination of outliers from the training set. This paper describes the OPF classifier in Section 4.2, introduces the new method

in Section 4.3, shows the experiments in Section 4.4 and Section 4.5 brings together the conclusions.

4.2 Optimum-path forest classifier

For the sake of completion, the OPF classifier will be briefly described. For more details, see [8].

Let Z_1 , Z_2 and Z_3 be the disjoint training, evaluating and testing sets, randomly chosen from a dataset Z , such that $Z = Z_1 \cup Z_2 \cup Z_3$. Let $\lambda(s)$ be a function which maps each sample $s \in Z$ to its correct class $i, i = 1, \dots, c$. Let $\vec{v}(s)$ denote the feature vector of sample s , computed by some feature extractor v and $d(s, t)$ denote the distance between s and t (e.g. euclidean distance).

A specially chosen subset $S \subset Z_1$, whose elements are called *prototypes*, contains the samples responsible for classifying each $t \in Z \setminus S$. This classification is accomplished by determining, for each t , which $s \in S$ is the most closely (strongly) connected to t and setting the class of t to be that of s .

The strategy of the OPF classifier is to first obtain an optimum discrete partition of the feature space of Z_1 , so that it becomes possible to classify every $t \in Z_2 \cup Z_3$ based on that partition, in an incremental way.

Let (Z_1, A) be the complete graph which represents the training set, where to each sample $s \in Z_1$ corresponds a node and such that to each edge $\langle s, t \rangle$ is assigned a cost equal to $d(s, t)$. Let $f(\pi_t^{s_1})$ be a function which evaluates the cost of the path $\pi_t^{s_1} = \langle s_1 = t_1, t_2, \dots, t_n = t \rangle$, starting at some $s_1 \in S$ and ending at $t \in Z_1$. A path π_t^s is considered optimum if, for any other path $\pi_t^{s'}$, $f(\pi_t^s) \leq f(\pi_t^{s'})$. The union of optimum-paths for all $t \in Z_1$ forms a forest (an acyclic graph).

The optimum path forest is computed using the Image Foresting Transform algorithm [6] by basically solving the following optimization problem:

$$C(t) = \min_{\forall s \in S} \{f_{max}(\pi_t^s)\} \quad (4.2.1)$$

which gives an optimum path-cost $C(t)$ for each $t \in Z_1$ of an optimum path, denoted π_t^* for short, which starts at some $s \in S$. The function $f_{max}(\pi_t^s)$ computes the maximum edge cost along the path π_t^s .

Algorithm 1 shows the general procedure to compute the Optimum-Path Forest [8].

Algorithm 1 – OPF-ALGORITHM

INPUT: Training set Z_1 , λ -labeled prototypes $S \subset Z_1$.
 OUTPUT: Optimum-Path Forest P (predecessor map), path-cost map C and label map L and a list Z'_1 of the training nodes ordered by their path-cost.
 AUXILIARY: Priority queue Q and cost variable $cost$.

1. **For each** $s \in Z_1 \setminus S$ **do** $C(s) \leftarrow +\infty$.
2. **For each** $s \in S$ **do**
3. $C(s) \leftarrow 0, P(s) \leftarrow nil$
4. $L(s) \leftarrow \lambda(s), insert\ s\ into\ Q$.

```

5. While  $Q$  is not empty do
6.    $\left[ \begin{array}{l} \text{Remove from } Q \text{ a sample } s \text{ with minimum } C(s) \text{ and insert } s \text{ in } Z'_1. \\ \text{For each } t \in Z_1 \text{ such that } C(t) > C(s) \text{ do} \\ \text{Compute } cst \leftarrow \max\{C(s), d(s, t)\}. \\ \text{If } cst < C(t) \text{ then} \\ \quad \left[ \begin{array}{l} \text{If } C(t) \neq +\infty \text{ then remove } t \text{ from } Q. \\ P(t) \leftarrow s, L(t) \leftarrow L(s) \text{ and } C(t) \leftarrow cst. \\ \text{Insert } t \text{ in } Q. \end{array} \right. \end{array} \right.$ 
12.

```

The training process (i.e., finding the set S of prototypes) consists of computing a Minimum Spanning Tree (MST) on Z_1 and, for each edge $\{s, t\}$ of the MST, if $\lambda(s) \neq \lambda(t)$ then s and t are marked as prototypes.

In the classification phase, for every sample $t \in Z_3$ all possible paths from each $s \in Z_1$ to t are computed and the optimum one, π_t^* , is chosen. This path can easily be identified by incrementally evaluating the optimum cost $C(t)$ as follows:

$$C(t) = \min_{\forall s \in Z'_1} \{\max\{C(s), d(s, t)\}\}. \quad (4.2.2)$$

The (first) role of Z'_1 is to speed up the evaluation of Equation 4.2.2 which can halt when $\max\{C(s), d(s, t)\} < C(p)$, for a node p whose position in Z'_1 succeeds the position of s [7].

The learning phase consists of performing, after the training process, the classification of each sample in Z_2 through the method just described and, subsequently, swapping each incorrectly classified sample from Z_2 for a randomly chosen sample from $Z_1 \setminus S$.

4.3 Optimum-Path Forest Classifier with Outliers Detection: OPF-OD

Outliers in the training set might be identified through two kinds of errors: when a sample is attributed an incorrect label (false positive - FP) and, when a classifying sample does not identify a sample of its own class (false negative - FN). Samples that cause such errors negatively impact the accuracy in the classification phase unless they are detected during the learning phase (through incorrect classifications) and properly dealt with, as will be described later.

Additionally, it becomes necessary to compute the number of correct classifications (true positives - TP) and of correct rejections (true negatives - TN) to be contrasted with the false positives and false negatives in order to deduce the usefulness of a sample in the training set.

To understand this bookkeeping, let $s \in Z'_1$ be the sample in Equation 4.2.2 which classifies a sample $t \in Z_2$.

- The false positive counter for s , FP_s , is incremented if $L(t) \neq \lambda(t)$, otherwise, the true positive counter for s , TP_s , is incremented.

Now, proceeding upwards (i.e., as $C(\cdot)$ decreases) on Z'_1 , let p_1 and p_2 be the first samples found such that $L(p_1) = \lambda(t)$ and $L(p_2) \neq \lambda(t)$.

- The false negative counter for p_1 , FN_{p_1} , and the true negative counter for p_2 , TN_{p_2} , are incremented.

After computing FP, TP, FN and TN for all samples in Z_1 , a penalty pen_s is calculated for each $s \in Z_1$ so that a sample s can be considered an outlier whenever $\text{pen}_s > \varepsilon$, for some threshold $0 < \varepsilon < 1$.

Let

$$E_{s_+} = \frac{\text{FP}_s}{\text{TP}_s} \quad E_{s_-} = \frac{\text{FN}_s}{\text{TN}_s},$$

where E_{s_+} and E_{s_-} are defined as *acceptance* and *rejection rates* for s , which are then normalized in ξ_{s_+} and ξ_{s_-} , respectively, by their maximum possible values.

Define pen_s as follows:

$$\text{pen}_s = \mu \cdot (\xi_{s_+}) + (1 - \mu) \cdot (\xi_{s_-}) \quad (4.3.3)$$

where $0 \leq \mu \leq 1$, and hence, $0 \leq \text{pen}_s \leq 1$.

In other words, this penalty corresponds to the convex combination of the normalized acceptance and rejection rates, weighted by a parameter μ which allows for a user-controlled balance between FP and FN, that can be application dependent.

Once the penalties are computed for all samples in Z_1 , any sample s identified as an outlier (i.e., $\text{pen}_s > \varepsilon$) is swapped for a random sample from Z_2 . However, care is taken to prevent s from being swapped back into Z_1 , if it happens to be misclassified in Z_2 during the learning phase (see Section 4.2). This will guarantee that previously identified outliers are immovable from Z_2 so as not to negatively impact the accuracy of the classification of Z_3 .

Algorithm 2 explains the proposed approach.

Algorithm 2 – OPF-LEARNING-WITH-OUTLIERS-DETECTION

INPUT: λ -labeled training and evaluating sets Z_1 and Z_2 , number T of iterations, and threshold ε .
 OUTPUT: Instance of the OPF-OD classifier with best accuracy over Z_2 .
 AUXILIARY: List LM of misclassified samples and Acc .

1. **For each** iteration $i \leftarrow 1, 2, \dots, T$ **do**
2. Compute S from Z_1 .
3. $(P, C, L, Z'_1) \leftarrow \text{Algorithm-1}(Z_1, S)$.
4. **For each** sample $t \in Z_2$ **do** $\triangleright \triangleright \triangleright$ classify all samples in Z_2
5. Use the classifier computed in Line 3 to classify t with label $L(t)$.
6. **If** $\lambda(t) \neq L(t)$ **then** $LM \leftarrow LM \cup t$. $\triangleright \triangleright \triangleright$ save missclassified samples
7. Update FP, TP, FN, TN for the corresponding samples (Section 4.3).
8. **For each** sample $s \in Z_1$ **do** $\triangleright \triangleright \triangleright$ compute all penalties
9. Compute pen_s using Equation 4.3.3
10. **If** $\text{pen}_s > \varepsilon$ **then** Swap s for a sample from Z_2 .

11. $\left[\begin{array}{l} \text{Compute accuracy } Acc \text{ and save the actual instance of the classifier if} \\ \text{its accuracy is maximum so far.} \\ k \leftarrow |Z'_1|. \\ \mathbf{While } LM \neq \emptyset \text{ and } k > 0 \mathbf{ do} \triangleright \triangleright \triangleright \text{ swap missclassified samples} \\ \quad \left[\begin{array}{l} \text{Choose a random sample } t \text{ from } LM, LM \leftarrow LM \setminus t \\ \mathbf{While } Z'_1[k] \text{ is a prototype and } k > 0 \mathbf{ do } k \leftarrow k - 1. \\ \mathbf{If } k > 0 \mathbf{ then Swap } t \text{ for sample } Z'_1[k]. \\ k \leftarrow k - 1 \end{array} \right. \end{array} \right.$
- 12.
- 13.
- 14.
- 15.
- 16.
- 17.
- 18.
19. **Return** instance of the classifier with highest accuracy.

The samples to be swapped from Z_1 in Line 10 are not randomly chosen as in [8], but in decreasing order of path-cost using Z'_1 , because samples with higher path-cost have less probability of correctly classifying samples from Z_2 . Therefore, this gives a third important role for the ordered list Z'_1 .

4.4 Experiments

In this section, the datasets used in the tests, the computational environment, the classifiers employed and the overall approach of the experiments are described.

4.4.1 Datasets used

To validate the proposed approach, the following datasets from the UC Irvine Repository [3] were used:

- The “Adult” dataset, with about 45000 samples, 14 attributes and 2 classes.
- The “King-Rook vs. King” dataset (krk), with about 30000 samples, 6 attributes and 18 different classes.
- The “Gamma telescope” dataset, with about 20000 samples, 11 attributes and 2 possible classes.

Also, the “Mpeg 7” dataset [1] was used, which consists of about 1400 samples, 70 classes and whose features were extracted using a shape descriptor called Bean Angle Statistics (BAS) [2]. The distance between feature vectors was measured using euclidean distance and optimal correspondence subsequence (OCS) which is more appropriate for the BAS descriptor.

The classifiers used were the OPF with the traditional learning algorithm and the proposed OPF-OD method which refines OPF.

The accuracy measure used to evaluate the classifiers is computed as follows [8]:

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

$$E(i) = e_{i,1} + e_{i,2} \quad Acc = 1 - \frac{\sum_{i=1}^c E(i)}{2c}$$

where $FP(i)$ and $FN(i)$ are the number of false positives and false negatives for class i and $Z_3(i)$ is the number of samples from class i which belong to Z_3 .



4.4.2 Sizes of Z_1 and Z_2

Due to the fact that OPF-OD is based on a learning phase, the evaluating set Z_2 must be significantly larger than the training set Z_1 to allow for the computation of meaningful penalties for each sample in Z_1 . To ensure a good trade-off between accuracy and learning time, the most beneficial ratio $|Z_2|/|Z_1|$ was experimentally determined to be 5.

4.4.3 Results

In this section, comparisons between OPF and OPF-OD are presented. All the experiments reported here were run on an Intel™ Xeon™ 4-core 2.50 GHz processor, with 8 GB of RAM, under Linux but *without* multi-thread execution.

Firstly, the Adult dataset was used to test 10 different sizes for the training set Z_1 averaged over 10 executions for each size. Figure 4.1 presents the accuracy obtained, showing the average value and the standard deviation for each size. Note that in *all* cases, OPF-OD obtained higher accuracy than standard OPF.

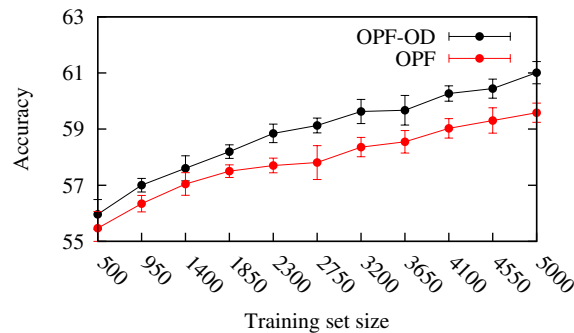


Figure 4.1: Accuracy for OPF and OPF-OD calculated over the classification of Z_3 using different sizes for Z_1 . Average of ten runs and standard deviation are shown. Here $|Z_2|/|Z_1| = 5$.

Considering the extra time required for the learning phase, the training time for OPF-OD was, as expected, higher than for OPF, as shown in Figure 4.2.(a) for the Adult dataset.

However, swapping out the outliers from Z_1 leads to some gain, time-wise, in the classification phase, as it can be seen in Figure 4.2.(b). This is due to the faster convergence of the optimization process that searches, for each sample in Z_3 , for its closest sample from Z_1 , as per Equation 4.2.2, since fewer samples have to be tested when outliers are no longer present.

Finally, Table 4.1 shows the accuracy and time results for the other datasets. Two different sizes for Z_1 (depending on the dataset) were considered. The accuracy of OPF-OD for these datasets was higher than OPF's, while OPF-OD's performance was slightly worse in the training phase and better in the classification phase.

Note that the accuracy grows higher for all datasets as the training set becomes larger. For instance, with a training set of 140 samples for the mpeg7-OCS dataset, the accuracy



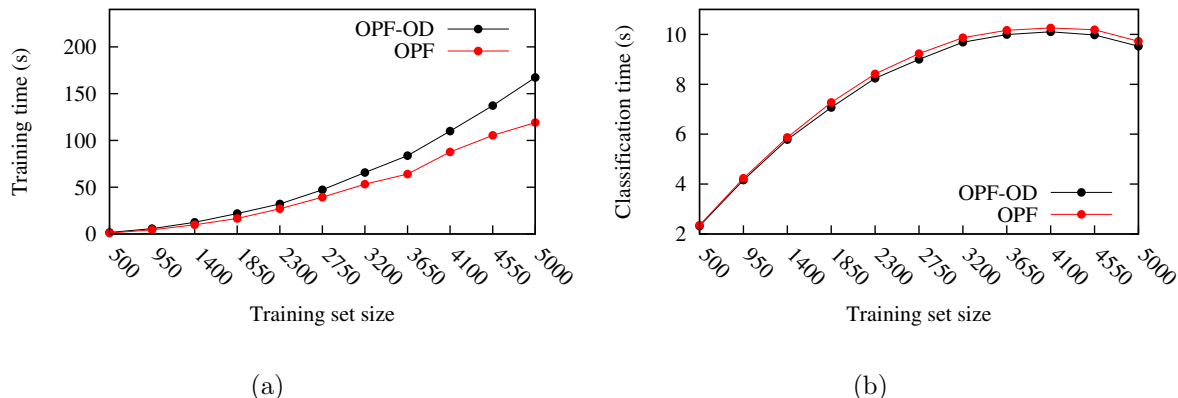


Figure 4.2: (a) Training time, and (b) Classification time in seconds for OPF and OPF-OD using different sizes for Z_1 . Average of ten runs is shown. Here $|Z_2|/|Z_1| = 5$.

was 11 percentage points higher than with half as many samples. Furthermore, it is critical to highlight that accuracy depends on other factors such as the distance function, as was the case of the mpeg7 dataset computed with euclidean distance vs. OCS distance.

Table 4.1: Accuracy (mean \pm standard deviation), Training time (mean) and Classification time (mean) in seconds for several datasets. Best values are in bold.

Datasets	$ Z_1 $	$ Z_2 $	$ Z_3 $	Accuracy		Training time		Classification time	
				OPF	OPF-OD	OPF	OPF-OD	OPF	OPF-OD
krk	280	1400	26320	58.66 \pm 0.64	58.90 \pm 0.90	0.155	0.328	0.468	0.460
krk	2800	14000	11200	69.24 \pm 0.47	67.68 \pm 0.54	26.773	29.367	1.976	1.915
gamma	190	950	17860	63.66 \pm 1.24	67.36 \pm 0.74	0.115	0.190	0.261	0.228
gamma	1900	9500	7600	67.24 \pm 0.72	70.74 \pm 0.75	11.589	15.997	1.109	0.982
mpeg7	70	350	980	68.09 \pm 1.76	71.53 \pm 1.02	0.039	0.105	0.0194	0.0192
mpeg7	140	700	560	78.09 \pm 1.49	79.68 \pm 1.49	0.145	0.348	0.0223	0.0221
mpeg7-OCS	70	350	980	75.57 \pm 0.82	78.40 \pm 1.65	0.013	0.039	0.0007	0.0008
mpeg7-OCS	140	700	560	89.93 \pm 0.66	90.92 \pm 0.65	0.022	0.065	0.0010	0.0011

4.5 Conclusions

In this work, a novel learning algorithm was presented that improves the time performance and accuracy of the supervised Optimum Path Forest classifier. This algorithm is essentially based on an ingenious method of detecting outliers in the training set and their subsequent swapping for new random samples from the evaluating set leading to a refined training set. As substantiation, notice that the attained improvement in accuracy was as high as 3.7 percentage points in the case of the Gamma dataset. This is mostly due to the

resulting samples in the training set being better representatives of the classes present in the testing set than if outliers had remained in it, causing negative impact in accuracy. Despite the fact that the new learning algorithm performs extra work to identify outliers, the overall training time is only slightly increased. However, for most applications, this is well compensated by the lower classification time which results from the faster convergence of the search process through the elements of the training set when outliers have been swapped out. Hence, the forest computed by OPF-OD over the training set is more effective and efficient than OPF's. For instance, for the Adult dataset, the behavior of training time vs. classification time is illustrated in Figures 4.2.(a) and 4.2.(b).

References

- [1] MPEG-7: The generic multimedia content description standard, part 1. *IEEE MultiMedia*, 9:78–87, 2002.
- [2] 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, 2003.
- [3] A. Asuncion and D.J. Newman. UCI machine learning repository. 2007.
- [4] 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.
- [5] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.
- [6] 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.
- [7] J.P. Papa, F.A.M. Cappabianco, and A.X. Falcão. Optimizing optimum-path forest classification for huge datasets. In *Proceedings of The 20th International Conference on Pattern Recognition*, 2010. (to appear).
- [8] J.P. Papa, A.X. Falcão, and C.T.N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19:120–131, 2009.
- [9] V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

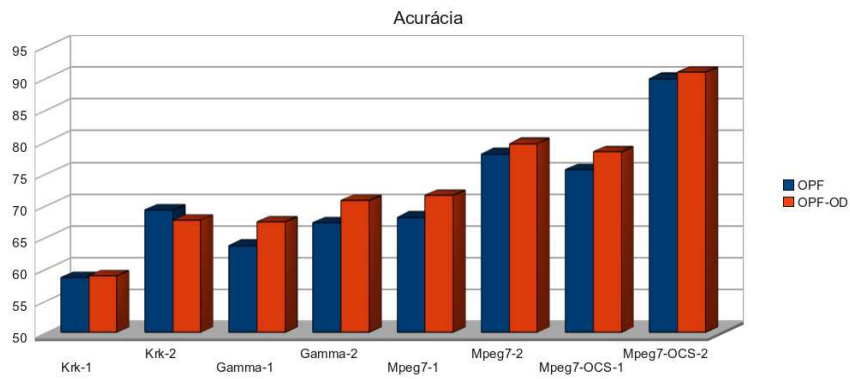
Comentários

C1: Domínio das Bases de Dados

As bases de dados utilizadas nos testes correspondem a diferentes domínios de estudo, e os atributos extraídos foram determinados segundo as características próprias do problema. A base “Adult” vem das ciências sociais, a base “K.R.K” é sobre predição de jogos e a base “Gamma” tem origem em problemas de astrofísica.

C2: Gráfico dos Resultados

Para se ter uma melhor idéia dos resultados do método OPF-OD sobre o restante das bases de dados, apresentam-se os resultados de acurácia da Tabela 4.1 no seguinte gráfico:



Capítulo 5

A Novel Outlier Detection Method based on the Optimum-Path Forest Supervised Classifier

Prólogo

O artigo incluído neste capítulo, em co-autoria com Pedro J. de Rezende e Alexandre X. Falcão, ambos do Instituto de Computação da Universidade Estadual de Campinas e João Paulo Papa, do Departamento de Ciência da Computação da Universidade Estadual Paulista, campus Baurú, foi submetido à Revista *Pattern Recognition*.

Este artigo constitui uma extensão do artigo apresentado no Capítulo 4, cujo objetivo é unicamente a detecção dos outliers, e os testes neste artigo foram conduzidos para tal tarefa, apresentando, por exemplo, uma curva ROC para mostrar a eficácia do algoritmo. O artigo apresenta uma revisão de algumas técnicas de detecção de outliers presentes na literatura, os conceitos fundamentais do Classificador OPF e depois os detalhes do novo algoritmo. Neste algoritmo, são extraídas duas medidas para cada amostra com o intuito de classificá-la como outlier, sendo que essas medidas são computadas de maneira diferente às do primeiro artigo, melhorando a sua taxa de acertos na detecção. Ademais, a metodologia de teste usada permite medir a capacidade do algoritmo em detectar erros de rotulação nos conjuntos de dados usados, a qual foi a motivação deste trabalho.

O método obteve resultados interessantes, mostrando-o eficaz e eficiente. As medidas usadas foram: Taxa de Falsos Positivos (FPR) e Taxa de Verdadeiros Positivos (TPR), que são as medidas usadas na curva ROC, sendo que a FPR é baixa para todos os casos e a TPR é alta na maioria dos casos. Por outro lado, o tempo de execução do método é muito baixo, o que se constitui numa das grandes vantagens do nosso método.

Abstract

Outliers may appear in unlabeled and labeled datasets, the last case being caused by errors during their automatic classification or manual annotation. Most methods for outlier detection rely on discrepancy measures computed within a neighborhood of each sample in the feature space. As the datasets grow larger, the efficiency of those methods becomes seriously compromised. We present here a novel algorithm which infers local discrepancy measures and detects outliers in labeled datasets with no need for expensive neighborhood computation. Our method exploits the fast optimum-path forest supervised classifier to detect outliers according to their performance and usefulness in classifying other samples. Furthermore, the way we define the outliers ground-truth lets us simulate labeling errors in the datasets, measuring also the ability of the method to retrieve these errors in order to alert the specialist. The method was found to be effective in detecting outliers, as measured both by the number of true positives and false positives, besides being efficient in computational time.

5.1 Introduction

An outlier is an observation which exhibits different characteristics than the rest of the observations in a dataset, and therefore, presents high probability of causing errors in procedures performed over those data, such as statistical analysis, image processing, etc. Consequently, it is imperative to have methods that allow successful detection of outliers, allowing one to work with more homogeneous data.

Several outlier detection techniques have been proposed in the literature, the most common and widely explored ones being based on statistical analysis. In general, the idea is to compute measures such as mean, median, standard deviation, and distance measures over the data in order to discover whether there exist samples significantly different from the rest, according to those measures. For instance, the Mahalanobis distance [1] is a favored approach in the literature, as it can give an estimate of how far from the majority of the data a sample is. Other classical methods include the class of M -estimators [2] (including the Monotone and the Redescending M -estimators), S -estimators [3], Combinatorial estimators [4], Compound estimators [5], density estimators [6], the BACON method [7], the nearest-neighbor approach [8], and methods based on active learning [9]. There are also methods based on the concept of *projection pursuit* [10]. Their goal is to pursue different projections of the data in which interesting structures will be revealed when viewed in lower dimensions, leading to a choice of the most interesting projection. Two well-known methods which use projection pursuit are the Stahel-Donoho Estimator [11] and the Kurtosis method [12].

However, the main drawback of existing outlier detection techniques is that they take into account some discrepancy measure within a neighborhood of each sample in the feature space or must solve global optimization problems for each sample, and for these reasons their computational efficiency becomes seriously compromised as datasets grow larger, coming to be inviable for high-dimensional spaces. Moreover, they usually aim to detect outliers in unlabeled datasets. A very interesting application, however, which is the motivation of our work, is the detection of misclassified samples or manual-labeling

errors in a dataset, i.e., errors caused by an automatic classification process or induced by a specialist during the creation of the ground-truth of the dataset. Depending on how we define the outliers in a dataset to test the method, it is possible to simulate those situations and to measure the ability of the method to detect those errors.

As stated by M. Werner [13], the most desirable features of a successful outliers detection method are its *sensitivity* and its *specificity*. The sensitivity is the ability to detect genuine outliers (true positives), and the specificity is the ability to not mistake regular samples for outliers (false positives). Also, it is desirable that the method be efficient, specially if we are working with large high-dimensional datasets.

In this work, we exploit a fast supervised learning algorithm based on optimum-path forest [14] to infer local discrepancy measures and detect outliers in labeled datasets, with no need for expensive neighborhood computation. Firstly, the method randomly splits the dataset into a smaller training set and a larger evaluating set, for the sake of efficiency, as required in the case of large datasets. The idea in this supervised learning algorithm is to project an Optimum-Path Forest classifier (OPF) from the training set and apply it on the evaluating set during a few iterations. In this process, misclassified evaluating samples are randomly swapped by *non-representative* training samples in order to improve the quality of the training set while increasing its accuracy in the classification of the evaluating set. The OPF classifier has a very low computational training cost, since it does not have to optimize parameters as other methods do. Furthermore, it can deal with multi-class data, since its formulation is based on multiple prototypes that best represent the various classes, as well as classes with a reasonable degree of overlapping.

The method we propose here uses measures extracted during the learning process of the OPF classifier to compute the *Relative Density* and the *Degree of Isolation* of each sample in the training set. It then uses these measures to define which samples are outliers based on some threshold. These outliers are then exchanged with new samples from the evaluating set, where they remain to no effect in all subsequent iterations. One advantage of our method is its low computational complexity, which makes it promising for outlier detection in large datasets. Due to its specificity, though, it should be remarked that this method is not applicable to unlabeled datasets.

This paper briefly describes the OPF classifier in Section 5.2, introduces the OPF-based outlier detection method in Section 5.3, discusses the testing methodology and empirical results in Section 5.4 and summarizes conclusions in Section 5.5.

5.2 Optimum-path forest classifier

The main goal of any pattern classification system is to obtain the best possible accuracy in predicting the class of new patterns [15]. The design of pattern classifiers is usually based on a training set with samples that should represent the distribution of the classes in the feature space. In this work, the supervised learning case is addressed, where the class of each training sample is known.

Several methods include a learning phase in order to improve the behavior of the classifier in predicting the classes of new data. This process is based on two labeled sets with training and evaluating samples, usually selected at random, where the evaluating

set is used as a pseudo-test for the classifier derived from the training set. Depending of the strategy used, this phase aims at improving the classification of the testing set.

Let Z_1 , Z_2 and Z_3 be disjoint training, evaluating and testing sets, randomly chosen from the dataset Z , such that $Z = Z_1 \cup Z_2 \cup Z_3$. Let $\lambda(s)$ be a function that gives the correct class $i \in \{1, \dots, c\}$, for each sample $s \in Z$. Let $\vec{v}(s)$ denote the feature vector of sample s (computed by some feature extractor v), and $d(s, t)$ denote the distance between s and t (computed with some distance function d). The goal of the OPF classifier is to obtain an optimum discrete partition of the feature space of Z_1 , as a representative set of Z , such that it becomes possible to classify every $s \in Z_2 \cup Z_3$ based on that partition. An specially chosen subset $S \subset Z_1$, whose elements are called prototypes (see Section 5.2.1), will be responsible for classifying each sample $t \in Z_1 \setminus S$, to obtain that optimum partition, as well as each sample $t \in (Z_2 \cup Z_3) \setminus S$ for evaluation or test. This will be accomplished by the method described below.

Let (Z_1, A) be the complete graph representing the training set, where to each sample $s \in Z_1$ corresponds a node and such that to each edge $\langle s, t \rangle$ is assigned a cost equal to $d(s, t)$. Let $f(\pi_t^{s_1})$ be a function which evaluates the cost of path $\pi_t^{s_1} = \langle s_1 = t_1, t_2, \dots, t_n = t \rangle$, starting at some $s_1 \in S$ and ending at $t \in Z_1$. A path π_t^s is considered optimum if, for any other path $\pi_t^{s'}$, $f(\pi_t^s) \leq f(\pi_t^{s'})$. The union of optimum paths for all $t \in Z_1$ forms a forest (an acyclic graph), called the Optimum Path Forest (OPF), each of whose trees is rooted at some sample $s \in S$.

The OPF is computed using the Image Foresting Transform (IFT) algorithm [16], with function $f = f_{max}$, for computing path cost defined by:

$$f_{max}(\langle s \rangle) = \begin{cases} 0 & \text{if } s \in S \\ +\infty & \text{otherwise} \end{cases}$$

$$f_{max}(\pi_{s'}^s \cdot \langle s', t \rangle) = \max\{f_{max}(\pi_{s'}^s), d(s', t)\} \quad (5.2.1)$$

where $\pi_{s'}^s \cdot \langle s', t \rangle$ denotes the concatenation of a path $\pi_{s'}^s$ with an edge $\langle s', t \rangle$, and $\langle s \rangle$ denotes a trivial path. Therefore, the function $f_{max}(\pi_t^s)$ computes the maximum edge cost along the path π_t^s .

Basically, the OPF algorithm solves the following optimization problem:

$$C(t) = \min_{\forall \pi_t^s \text{ in } (Z_1, A)} \{f_{max}(\pi_t^s)\} \quad (5.2.2)$$

which gives an optimum-path cost $C(t)$ for each $t \in Z_1$, setting the optimum path π_t^* , which starts at some $s \in S$.

Figure 5.1 shows the resulting forest from a training set Z_1 , according to Equation 5.2.2.

Algorithm 3 shows the general procedure to compute the Optimum-Path Forest [14].

Algorithm 3 – OPF-ALGORITHM

INPUT: Training set Z_1 , λ -labeled prototypes $S \subset Z_1$.
 OUTPUT: Optimum-Path Forest P (predecessor map), path-cost map C , label map L and a list Z_1' of the training nodes ordered by their path-cost.
 AUXILIARY: Priority queue Q and cost variable cst .

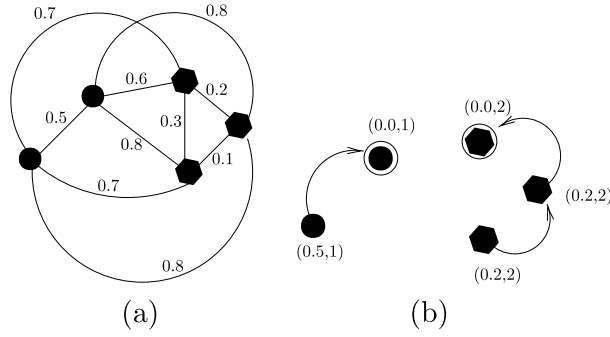


Figure 5.1: (a) Complete graph (Z_1, A) for training set Z_1 . (b) Optimum-Path Forest for (Z_1, A) . (Extracted from [14]).

1. **For each** $s \in Z_1 \setminus S$ **do** $C(s) \leftarrow +\infty$.
2. **For each** $s \in S$ **do**
3. $C(s) \leftarrow 0, P(s) \leftarrow nil$
4. $L(s) \leftarrow \lambda(s)$, *insert s into Q*.
5. **While** *Q is not empty* **do**
6. *Remove from Q a sample s with minimum C(s) and insert s in Z'_1.*
7. **For each** $t \in Z_1$ *such that* $C(t) > C(s)$ **do**
8. $cst \leftarrow \max\{C(s), d(s, t)\}$.
9. **If** $cst < C(t)$ **then**
10. **If** $C(t) \neq +\infty$ **then** *remove t from Q.*
11. $P(t) \leftarrow s, L(t) \leftarrow L(s)$ *and* $C(t) \leftarrow cst$.
12. *Insert t in Q.*

5.2.1 Training

The training procedure of the OPF method first consists of finding the set of prototypes S within the set Z_1 . As proposed in [14], this is done by computing a Minimum Spanning Tree (MST) on Z_1 , and for each edge (s, t) in the MST with $\lambda(s) \neq \lambda(t)$, set s and t as prototypes of their classes, $\lambda(s)$ and $\lambda(t)$, respectively.

Since the total cost of the edges on an MST is minimum over all other spanning trees over Z_1 , and since f_{max} is used, the MST contains an optimum path for every $t \in Z_1$. The choice of prototypes among the closest samples between classes aims to avoid optimum paths from distinct classes during classification. Training is concluded by executing Algorithm 3.

Figure 5.2 illustrates the training procedure based on the MST approach.

5.2.2 Classification

As proposed in [14], the classification procedure of a new node $t \in Z_2 \cup Z_3$ must consider incrementally all possible paths $\pi_t^* = \pi_{s'}^* \cdot \langle s', t \rangle$, such that $\pi_{s'}^*$ is an optimum prefix rooted in S with terminus in $s' \in Z_1$ in order to choose the optimum path with terminus t

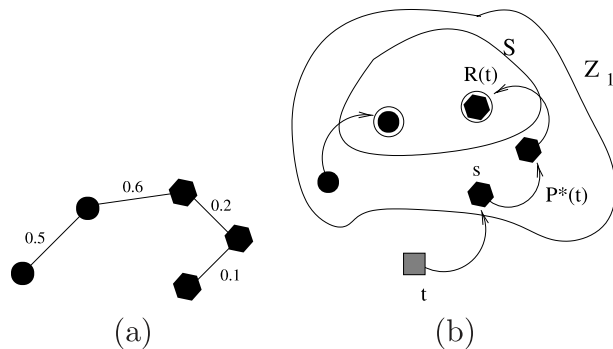


Figure 5.2: (a) MST for the complete graph (Z_1, A) of Figure 5.1, (b) Set of prototypes S chosen from the MST. (Extracted from [14]).

according to the equation below.

$$C(t) = \min_{\forall s' \in Z'_1} \{\max\{C(s'), d(s', t)\}\}. \tag{5.2.3}$$

Let $s \in Z_1$ be the training node that satisfies this equation and $R(s) \in S$ be its root node, such that $\pi_t^{R(s)}$ is the optimum path. By Algorithm 3 $L(s) = \lambda(R(s))$ and so $L(t)$ is assigned to the class label $L(s)$.

The (first) role of Z'_1 is to speed up the evaluation of Equation 5.2.3 which can halt when $\max\{C(s), d(s, t)\} < C(p)$, for a node p whose position in Z'_1 succeeds the position of s [17].

Figure 5.3 shows the classification procedure described here.

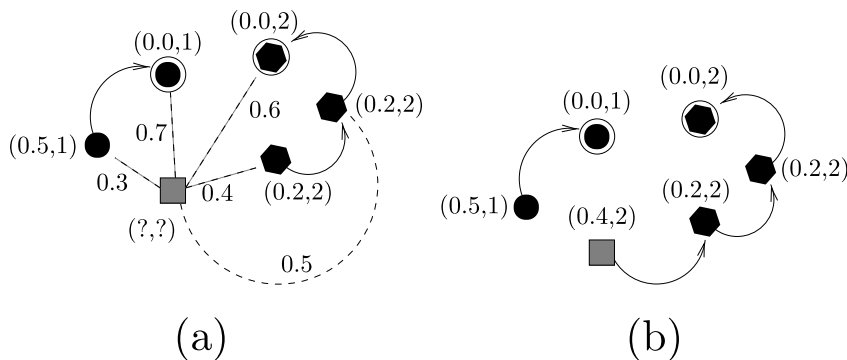


Figure 5.3: Classification of a new sample. (a) Computation of all the possible paths from s to Z_1 , (b) The optimum one π_t^* is chosen. (Extracted from [14]).

5.2.3 Learning

In [14], a learning algorithm was proposed in order to improve the accuracy of the OPF method for large datasets, yet preserving its high efficiency, such that the best samples

for the smaller set Z_1 can be learned from the larger set Z_2 . This algorithm consists of performing, after the training phase, the classification of Z_2 based on Equation 5.2.3 and, subsequently, swapping incorrectly classified samples from Z_2 for randomly chosen samples from $Z_1 \setminus S$. As this process is iterated, it is expected that an improvement in accuracy will follow since Z_1 should contain better representatives of the classes.

5.3 OPF-based Outlier Detection: OPF-OD

In statistics, an outlier is an observation that is numerically distant from the rest of the data, in other words, outliers may be indicative of data points that belong to a different class than the rest of the sample set. In pattern recognition, one can restrict this analysis to a neighborhood of a given sample in order to decide whether it is an outlier based on local information. Intuitively, a sample s can be considered an outlier whenever its vicinity $N(s)$ is comprised mostly of samples from classes different from $\lambda(s)$, the class of s . Topologically speaking, one would look at the ratio between the number of elements in $N(s)$ that belong to the class $\lambda(s)$, namely $N_\lambda(s) = \{t \in N(s) | \lambda(t) = \lambda(s)\}$ and that of $N(s)$. These values are hard or impossible to compute without analyzing the vicinity of s , i.e. it would be necessary to compute the nearest neighbors for each s , increasing the learning time. However, they can be estimated using some measures extracted during the learning process of the OPF classifier, without the need of explicitly computing that vicinity. We now describe how this can be achieved.

During the learning phase, as a sample s from the training set Z_1 is employed in the classification of elements of the learning set Z_2 , two types of errors can be imputed to s : when s attributes an incorrect label to a sample (false positive - FP) and when s does not identify a sample of its own class (false negative - FN). Additionally, it is useful to compute the number of correct classifications (true positives - TP) and of correct rejections (true negatives - TN) to be contrasted with the false positives and false negatives in order to deduce the usefulness of s in the training set.

To understand this bookkeeping, let $s \in Z_1'$ be the sample in Equation 5.2.3 which classifies a sample $t \in Z_2$.

- The false positive counter for s , FP_s , is incremented if $L(t) \neq \lambda(t)$, otherwise, the true positive counter for s , TP_s , is incremented.

Now, proceeding upwards on Z_1' (i.e., as $C(\cdot)$ decreases), consider a window W_s of k elements $\{p_1, p_2, \dots, p_k\}$ in Z_1' . We choose $k = 0.1 \times |Z_1|$ so that penalties are not spread through a large set of elements.

- The false negative counter for p_i , FN_{p_i} , is incremented if $L(p_i) \neq \lambda(p_i)$, otherwise, the true negative counter for p_i , TN_{p_i} , is incremented.

After computing FP, TP, FN and TN for all samples in Z_1 , we can estimate the number of elements in $N(s)$ and $N_\lambda(s)$ as follows:

$$|N(s)| = TP(s) + FN(s) + TN(s) + FP(s)$$

$$|N_\lambda(s)| = \text{TP}(s) + \text{FN}(s) \quad (5.3.4)$$

According to the definitions of TP, FN, TN and FP, these estimatives are correct, since we define that the elements in $N_\lambda(s)$ are the ones which were correctly classified by s ($\text{TP}(s)$) plus the ones that s did not classify but are from same class as s ($\text{FN}(s)$). For the case of $N(s)$, we consider all the elements classified, correctly or not, by s ($\text{TP}(s)$ and $\text{FP}(s)$) since they are the closest to s according to the function f_{max} , and all the elements not classified by s but that are close to s according to the ordered list Z'_1 ($\text{TN}(s)$ and $\text{FN}(s)$). Notice that the list Z'_1 is used to define which elements will be imputed with false negatives or true negatives. The rationale for this is that for each sample $s \in Z'_1$ the elements which would classify s , had its classifying sample not been present, lie immediately above s in Z'_1 . Hence, the elements which are in the window W_s are the ones closer to s according f_{max} and it is natural to consider that s is part of their vicinity.

To decide whether a sample $s \in Z_1$ is an outlier, we will define the *Relative Density* of its neighborhood $N(s)$, denoted $\text{RD}(s)$, and its *Degree of Isolation*, $\text{DI}(s)$, within that neighborhood, as follows:

$$\begin{aligned} \text{RD}(s) &= |N_\lambda(s)|/|N(s)| \\ \text{DI}(s) &= 1 - \text{RD}(s) \end{aligned} \quad (5.3.5)$$

Once DI is computed for all samples in Z_1 , any sample s is identified as an outlier if $\text{DI}(s) > \varepsilon$, for some threshold $0 < \varepsilon < 1$. In this case, s is swapped for a sample from Z_2 . However, care is taken to prevent s from being swapped back into Z_1 , whenever it happens to be misclassified in Z_2 during the iterations of the learning phase (see Section 5.2.3). This will guarantee that previously identified outliers are immovable from Z_2 so as to make it possible to evaluate all the samples in Z_2 . Moreover, when a sample $s \in Z_1$ is swapped into Z_2 , all samples in the (sub-)tree rooted at s become unclassified for the remaining of the current iteration. This is expected to cause no undesirable impact on the learning process since during the next iteration one of these samples is likely to classify most if not all of them.

Since the value of DI is defined for every sample in Z_1 , the probability of prototypes themselves being swapped out of Z_1 is low, as classifications are generally set off by regular samples and rarely directly by prototypes. This fact leads to a more stable set of prototypes throughout the iterations of the learning process while inducing a higher number of misclassifying samples being swapped out. This drives the algorithm towards having a better set of discriminating prototypes.

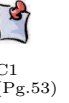
5.3.1 Defining the outliers in the training set

Selecting which samples will be marked as outliers is an essential part of this approach. A criteria based on the assumption that outliers will have high values of DI, while the non-outliers will have the low values is a promising one as long as we carefully choose the value of ε . In this way, we can maximize the number of outliers correctly identified while minimizing the number of false positives.

In this work, we explore two approaches to determine the best value of ε , namely, ROC curves and a dynamic method based on the distribution of the values of DI. The idea is

to find two 1-D clusters, one at one extreme of the distribution, where most of the outliers will be found, and the other at the opposite extreme comprised mainly of non-outliers. The cluster identification is attained as follows.

Let μ_{DI} be the average value of $\{DI(s) \mid \forall s \in Z_1\}$ and σ_{DI} the corresponding standard deviation. The set of candidates for outliers from Z_1 is initially set to $OC = \{s \in Z_1 \mid DI(s) > \mu_{DI} + \sigma_{DI}\}$. Then, scanning the samples in OC in increasing order of DI 's, we compute the gap $G(s_i, s_{i+1}) = DI(s_{i+1}) - DI(s_i)$ for all $s_i \in OC$, its respective mean μ_G and standard deviation σ_G . At this point, we have three natural choices for defining a splitting gap: i) look in increasing order of DI 's, searching for the left-most gap, say, of samples s_j and s_{j+1} , such that $G(s_j, s_{j+1}) > \mu_G + \sigma_G$; or ii) similarly, examine in decreasing order of DI 's, seeking the right-most gap with this property; or iii) choose the largest gap in OC . Next, define the threshold ε as the middle point of the chosen gap. These three choices will be duly analyzed in section 5.4.4.



5.3.2 Swapping outliers and wrongfully classified samples from the evaluating set

A sample $s \in Z_2$ to be swapped for an outlier is not chosen at random among all the samples in Z_2 , but only among the samples of its own class. Of course, this might not be possible for all the outliers to be swapped, as the available samples in Z_2 (i.e., samples that have not been in Z_1 previously) decrease throughout the iterations. When this happens, a sample s is chosen at random from an arbitrary class.

Besides swapping out the outliers from Z_1 to Z_2 , we also consider the original approach of swapping the misclassified samples from Z_2 for random non-prototypes samples from Z_1 , because, as Papa et. al [14] proposed, this improves the training process. In this work, we develop this idea further by choosing the samples from Z_1 to be swapped, not randomly, but in decreasing order of path-cost. The rationale for this is that samples from Z_1 with higher path-cost are less likely of being accountable for the classification of samples from Z_2 (see Equation 5.2.3). Therefore, this gives a third important role for the ordered list Z'_1 .

5.3.3 Proposed algorithm

Algorithm 4 – OPF LEARNING WITH OUTLIERS DETECTION

INPUT: λ -labeled training and evaluating sets Z_1 and Z_2 , number T of iterations.
 OUTPUT: Set OS , containing the outliers in $Z_1 \cup Z_2$.
 AUXILIARY: List LM of misclassified samples.

1. **For** iteration $i \leftarrow 1, 2, \dots, T$ **do**
2. $LM \leftarrow \emptyset$.
3. Compute S from Z_1 .
4. $(P, C, L, Z'_1) \leftarrow \text{Algorithm-3}(Z_1, S)$.
5. $\triangleright \triangleright \triangleright$ classify all samples in Z_2 .
6. **For each** sample $t \in Z_2$, **do**
7. Use the classifier computed in Line 4 to classify t with label $L(t)$.
8. $\triangleright \triangleright \triangleright$ save missclassified samples.

```

9.   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
10.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
11.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
12.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
13.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
14.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
15.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
16.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
17.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
18.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
19.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
20.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
21.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
22.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
23.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
24.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
25.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
26.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
27.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

```

If $\lambda(t) \neq L(t)$ **then**
 $LM \leftarrow LM \cup \{t\}$.
 Update FP, TP, FN and TN for the corresponding samples (Section 5.3).
▷▷▷ compute all DI's, ε and swap outliers.
For each sample $s \in Z_1$, **do**
 Compute $DI(s)$ according to Equation 5.3.5.
 Compute ε according to Section 5.3.1.
For each sample $s \in Z_1$, **do**
 If $DI(s) > \varepsilon$, **then**
 $OS \leftarrow OS \cup \{s\}$.
 Swap s for a sample from Z_2 (Section 5.3.2).
 $k = |Z'_1|$.
▷▷▷ swap missclassified samples.
While $LM \neq \emptyset$ and $k > 0$, **do**
 Choose a sample t from LM .
 $LM \leftarrow LM \setminus t$.
 Look for a non-prototype sample $Z'_1[k]$, in decreasing order of $C(\cdot)$.
 Swap t for sample $Z'_1[k]$.
Return OS .

5.4 Experiments

In this section, the overall approach of the experiments, the datasets used in the tests, the computational environment and the results are described.

All the experiments reported here were performed on an Intel[®] Xeon[®] 4-core 2.50 GHz processor, with 8 GB of RAM, under Linux Gentoo 2.6.31-r10 and without multi-thread execution.

5.4.1 Definition of outlier

According to theoretical definitions in statistics, a sample is an outlier if it belongs to a distribution different from the rest of the data. However, defining outliers for empirical evaluation purposes is a difficult task, since we often do not know the distribution of the data. For this reason, the most common way to test outlier detection methods is to use artificial distributions. Nevertheless, it is possible to use real datasets for our purpose, as Abe et. al [9] and Lazarevic and Kumar [18] have shown. They argue that outliers can be introduced in a dataset of know classes as a sparse class of events within the dataset (e.g., a very small class, samples with changed labels, etc.).

Furthermore, as stated in Section 5.1, this work aims at the detection of labeling errors, caused by some automatic classification procedure or incurred by a specialist during the creation of the ground-truth of the dataset. A way to simulate these situation is to randomly choose a few samples from the dataset and replace their label (e.g., by choosing any of the other labels at random) and considering them as outliers to be identified. Hence, after the execution of the method, we can to measure the ability of the method to identify these artificially created labeling errors.

Notice that the method we are proposing uses the supervised classifier only to detect the outliers in a labeled dataset. Therefore, we are not concerned about the classification accuracy of the method on Z_3 , but only with its ability to detect outliers in $Z_1 \cup Z_2$. In this sense, a given labeled dataset must be randomly split into a smaller training set Z_1 and a larger evaluating set Z_2 , while Z_3 can be empty.

5.4.2 Datasets

To validate the proposed approach, we have chosen datasets obtained from the University of California at Irvine’s Machine Learning Repository [19] belonging to different fields of study: the “Shuttle” dataset (aerospatial technology), the “King-Rook vs. King” dataset (game prediction), the “Gamma Telescope” dataset (astrophysics), the “Nursery” dataset (medicine), the “Multi-Feat” dataset (text processing), the “Balance” dataset (experimental psychology) and the “Haberman” dataset (medicine). Table 5.1 shows the number of samples, classes and attributes of each dataset.

Table 5.1: Datasets used in the tests.

Dataset	Numb. Samples	Numb. Features	Numb. Classes
Shuttle	58000	9	7
K.R.K.	28056	6	18
Gamma	19020	10	2
Nursery	12960	8	5
Multi-Feat	1988	649	10
Balance	625	4	3
Haberman	306	3	2

5.4.3 Efficiency in training

Since we are using a supervised classifier approach to detect outliers, the most natural methodology would be to compare its performance against that of other classifiers, such as Support Vector Machines (SVM), Artificial Neural Networks (ANN), etc., configured to detect outliers as our method does. However, SVM and ANN spend way too much computational time in comparison with OPF, since they must perform several optimizations in higher dimensions during their learning phases. Papa et. al [14] have compared the OPF classifier with SVM (using the RBF kernel) and ANN (using the Multi-Layer Perceptron algorithm) for classification tasks, showing that OPF was about 70 times and 400 times faster than SVM and ANN, respectively. As further evidence of this disparity, we have compared the training times of the OPF, OPF-OD and SVM (with RBF and linear kernels) using one of the datasets chosen for our tests (Figure 5.4). Notice that for the largest Z_1 , SVM-RBF took about 6000 seconds and SVM-linear took about 1000, while OPF and OPF-OD required only around 100 seconds.

The majority of the methods we have found in the literature that use the classification approach spend much more computational time than OPF. For example, the method

proposed by Abe et. al [9] uses an ensemble of classifiers during the learning process to detect the outliers, which means that they repeat the entire process through a number of iterations. Lastly, methods based on density estimations [6] or nearest neighbors [8] are also more expensive than OPF, since they are very computational intensive.

Moreover, we are proposing a method to detect labeling errors in datasets which probably contain several thousand samples. Hence, it would not be viable to use any approach involving a classification method that requires a large number of optimizations, or a set of classifiers to solve the problem. For these reasons, we restricted the tests to the OPF classifier.

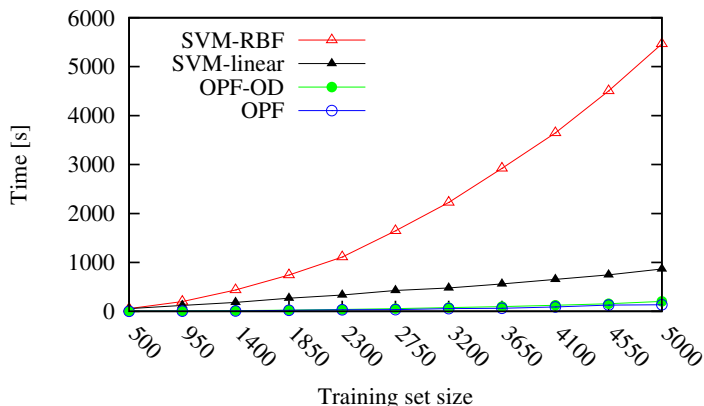


Figure 5.4: Training time for OPF, OPF-OD and SVM using the Shuttle dataset with various training set sizes ($|Z_2|$ was $5 \times |Z_1|$ for OPF and OPF-OD). Here, the OPF-OD was used to detect outliers and the others for an ordinary classification task.

5.4.4 Efficiency of the outlier detection task

As described before, we have used ROC curves to measure the performance of our method and we have also proposed a dynamic method that to compute the value of the threshold. The dynamic method has three possible configurations depending on how we choose the gap that splits the clusters of outliers and non-outliers (Section 5.3.1): i) the left-most gap approach (Dynamic-1), ii) the right-most gap approach (Dynamic-2), and iii) the largest gap approach (Dynamic-3).

Due to the fact that OPF-OD is based on the learning phase, the evaluating set Z_2 must be larger than the training set Z_1 for the sake of efficiency, as required by large datasets. On the other hand, Z_2 should have enough samples to allow for the computation of meaningful penalties for each sample in Z_1 . Hence, each dataset Z was randomly divided into Z_1 and Z_2 , containing 30% and 70% of the samples, respectively.

Table 5.2 shows the results for the different datasets. For each dataset, the optimum threshold was determined using the ROC curve. Then, the method was executed 15 times using the optimum threshold. Additionally, using the same partitions of the dataset, the three dynamic methods were executed 15 times. As the table shows, for some datasets

our method is able to obtain a reasonable balance between FP and TP, and for several of them, such as the Shuttle dataset, the performance is almost perfect. Furthermore, in some cases the results obtained by the dynamic method are as good as with the optimum threshold established using the ROC curve. To illustrate this, Figure 5.5 shows the results for the Shuttle dataset. Since this dataset provided good information about the method, we chose to display each point in the ROC curve as the mean of 15 executions. Notice that the ROC curve does not reach point (1,1), which means that the method does not need to consider all samples as outliers to retrieve all the true ones.

Table 5.2: False positive rate (FPR), True Positive Rate (TPR), Accuracy and Time for the datasets used in the tests. For each dataset we present the results for the optimum ε , chosen from the ROC curve, and the results for the best dynamic method (both chosen considering the highest accuracy). Accuracy is traditionally defined as $(TP + TN)/(TP + FP + TN + FN)$.

Dataset	Samples	Method	Optimum ε	FPR [0, 1]	TPR [0, 1]	Accuracy [0, 1]	Time [seconds]
Shuttle	58000	ROC analysis	0.93	0.11 ± 0.02	0.94 ± 0.04	0.91 ± 0.02	1097.56 ± 257.84
		Dynamic-3	-	0.10 ± 0.06	0.88 ± 0.14	0.89 ± 0.05	1105.17 ± 313.85
K.R.K.	28056	ROC analysis	0.99	0.14 ± 0.01	0.49 ± 0.01	0.67 ± 0.01	145.96 ± 3.68
		Dynamic-2	-	0.09 ± 0.01	0.32 ± 0.01	0.61 ± 0.01	148.03 ± 4.21
Gamma	19020	ROC analysis	0.52	0.37 ± 0.01	0.79 ± 0.01	0.71 ± 0.01	59.65 ± 4.95
		Dynamic-2	-	0.18 ± 0.01	0.57 ± 0.01	0.70 ± 0.01	76.08 ± 5.22
Nursery	12960	ROC analysis	0.96	0.06 ± 0.00	0.69 ± 0.01	0.81 ± 0.00	33.19 ± 4.65
		Dynamic-3	-	0.06 ± 0.01	0.69 ± 0.02	0.81 ± 0.01	33.55 ± 4.47
Multi-Feat	1988	ROC analysis	0.97	0.27 ± 0.04	0.78 ± 0.11	0.75 ± 0.05	19.14 ± 5.31
		Dynamic-3	-	0.03 ± 0.01	0.13 ± 0.08	0.55 ± 0.04	16.79 ± 4.11
Balance	625	ROC analysis	0.98	0.18 ± 0.02	0.53 ± 0.09	0.68 ± 0.04	0.70 ± 0.31
		Dynamic-3	-	0.14 ± 0.03	0.48 ± 0.06	0.67 ± 0.02	0.63 ± 0.19
Haberman	306	ROC analysis	0.93	0.40 ± 0.15	0.93 ± 0.25	0.77 ± 0.12	0.10 ± 0.00
		Dynamic-3	-	0.06 ± 0.05	0.13 ± 0.34	0.54 ± 0.15	0.63 ± 0.31

One of the greatest advantages of the OPF-OD method is its computational efficiency in detecting outliers. Table 5.2 shows the execution time of the method and the sizes of the datasets.

5.5 Conclusions

In this work, we have presented a novel outlier detection algorithm using the supervised learning process of the optimum-path forest classifier. This algorithm is a simple and yet effective method, which computes the *Relative Density* and *Degree of Isolation* for training samples based on their correct and incorrect classifications of evaluating samples and with no need for expensive neighborhood computations in the feature space. These measures enable us to determine, by thresholding, which training samples are outliers and, as the training samples are renewed during subsequent iterations of the learning phase, the method progressively detects outliers in the entire dataset.

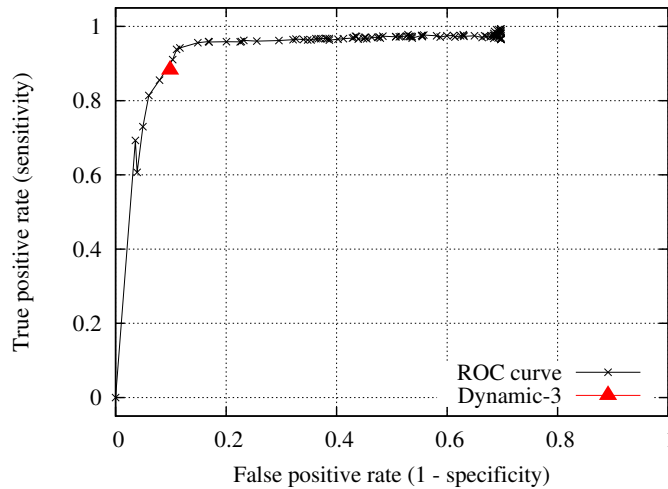


Figure 5.5: ROC curve and better dynamic method for the Shuttle dataset. Here $|Z_1| = 30\%$ and $|Z_1| = 70\%$ of $|Z|$.

The experimental methodology used indicates that the method is able to solve the interesting problem of identifying labeling errors produced in a dataset by automatic classification or by a specialist during the creation of its ground-truth. The results showed that our method is able to identify those errors with high specificity and high sensitivity, as Figure 5.5 shows for the Shuttle dataset.

Furthermore, the proposed method for dynamically computing the threshold for outlier detection proved effective when compared with the optimum threshold obtained using the (far more costly) ROC curve. Regarding the computational time needed for the detection of outliers, the results showed that the method is very efficient. For instance, in the case of the Nursery dataset, containing about 13000 samples, the overall process was performed in no more than 30 seconds.


C4
(Pg.54)

Acknowledgments

C. Castelo-Fernández was supported by CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Grant # 01-P-04388/2010.

Pedro J. de Rezende was partially supported by CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grants # 472504/2007-0, 483177/2009-1 and FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo – Grant #07/52015-0 and a Grant from FAEPEX/UNICAMP.

Alexandre X. Falcão was partially supported by CNPq Grants # 481556/2009-5 (ARPIS), 302617/2007-8 and FAPESP Grant #2007/52015-0.

João Paulo Papa was partially supported by CNPq Grant # 481556/2009-5 (ARPIS) and FAPESP Grant # 2009/16206-1.

References

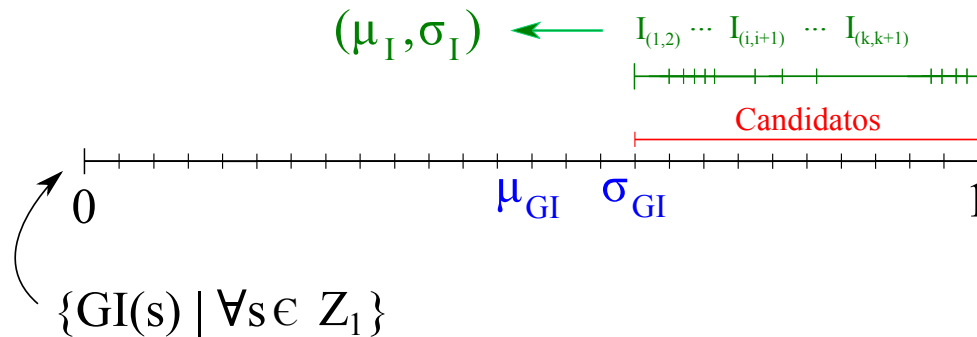
- [1] P.C. Mahalanobis, On the generalised distance in statistics, in *Proceedings National Institute of Science*, India, Vol. 2, 1936, pp. 49–55.
- [2] N. Campbell, Robust procedures in multivariate analysis i: Robust covariance estimation, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 29 (3) (1980) 231–237.
- [3] P. Rousseeuw, V. Yohai, Robust regression by means of s-estimators, *Robust and Nonlinear Time Series Analysis*, edited by J. Franke, W. Hardle and D. Martin, Lecture Notes in Statistics 26 (1984) 256–272.
- [4] P. Rousseeuw, Multivariate estimation with high breakdown point, *Mathematical Statistics and Applications*, edited by W. Grossmann, G. Pflug, I. Vincze, and W. Wertz B (1985) 283–297.
- [5] D. Woodruff, D. Rocke, Computable robust estimation of multivariate location and shape in high dimension using compound estimators, *Journal of the American Statistical Association* 89 (427) (1994) 888–896.
- [6] M.M. Breunig, H.P. Kriegel, R.T. Ng, J. Sander, Identifying density based local outliers, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.
- [7] A. Hadi, N. Billor, P. Velleman, BACON: Blocked adaptive computationally-efficient outlier nominators, *Computational Statistics and Data Analysis* 34 (3) (2000) 279–298.
- [8] E. Knorr, R. Ng, Algorithms for mining distance-based outliers in large datasets, in: *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 392–403.
- [9] N. Abe, B. Zadrozny, J. Langford, Outlier detection by active learning, in: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06)*, ACM, New York, NY, USA, 2006, pp. 504–509.
- [10] J. Friedman, J. Tukey, A projection pursuit algorithm for exploratory data analysis, *IEEE Transactions on Computers* 23 (9) (1974) 881–890.
- [11] D. Donoho, Breakdown properties of multivariate location estimators, Ph.D. thesis, Harvard University (1982).
- [12] D. Peña, F. Prieto, Multivariate outlier detection and robust covariance matrix estimation, *Technometrics* 43 (3) (2001) 286–310.
- [13] M. Werner, Identification of multivariate outliers in large datasets, Ph.D. thesis, University of Colorado at Denver (2003).

- [14] J. Papa, A. Falcão, C. Suzuki, Supervised pattern classification based on optimum-path forest, *International Journal of Imaging Systems and Technology* 19 (2009) 120–131.
- [15] R. Duda, P. Hart, D. Stork, Pattern Classification, 2nd Edition, Wiley-Interscience, 2000.
- [16] A. Falcao, J. Stolfi, R. Lotufo, The image foresting transform: Theory, algorithms, and applications, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (1) (2004) 19–29.
- [17] J.P. Papa, F.A.M. Cappabianco, A.X. Falcão, Optimizing optimum-path forest classification for huge datasets, in: *Proceedings of The 20th International Conference on Pattern Recognition*, IEEE Computer Society, 2010, pp. 4162–4165.
- [18] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '05)*, ACM, New York, NY, USA, 2005, pp. 157–166.
- [19] A. Asuncion, D. Newman, UCI machine learning repository (2007), URL www.ics.uci.edu/~mllearn/MLRepository.html (Access date: August 30th 2010).

Comentários

C1: Determinação dinâmica do limiar de Grau de Isolamento

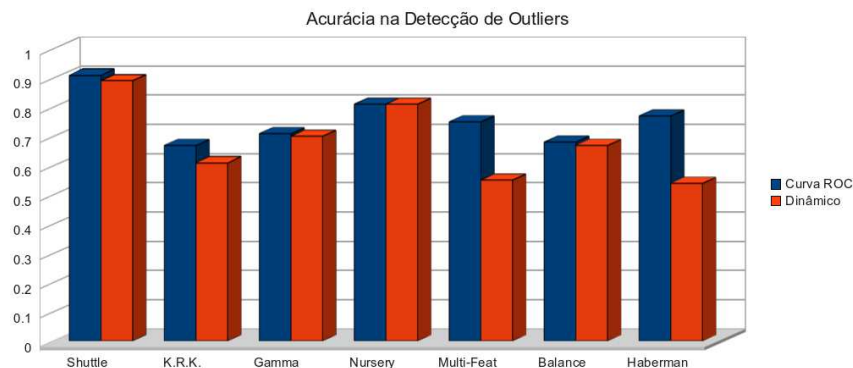
O método dinâmico para a detecção automática do limiar de Grau de Isolamento (GI), pode ser melhor explicado utilizando-se a seguinte figura:



Primeiro, computa-se a média μ_{GI} e desvio padrão σ_{GI} dos GI, definindo como candidatos os que superam o desvio padrão. Para esses candidatos, computam-se os intervalos I entre valores consecutivos e a sua respectiva média μ_I e desvio padrão σ_I . Finalmente, utilizando μ_I e σ_I , definem-se os três métodos dinâmicos: o primeiro maior que o desvio padrão pela direita, pela esquerda e o maior de todos.

C2: Gráfico dos Resultados

Para se ter uma melhor idéia dos resultados do método OPF-OD aprimorado sobre o restante das bases de dados, apresentam-se os resultados de acurácia na detecção de outliers da Tabela 5.2 no seguinte gráfico:



C3: Resultados da base Gamma

Como a Tabela 5.2 mostra, o valor do limiar ótimo ε para a base Gamma é bem diferente do restante das bases. Para se poder explicar tal comportamento, o melhor seria fazer uma inspeção visual da base, para conhecer a distribuição das amostras e dos possíveis

outliers. Porém, isso é inviável devido a os vetores de características da base terem 10 dimensões, a menos que se utilizem técnicas de visualização científica; mas esse não é o objetivo deste trabalho.

Uma possível explicação para aquele comportamento é a presença de concentrações de amostras da mesma classe na base e, ao mesmo tempo, um alto grau de superposição das classes. Portanto, as amostras que foram definidas como outliers na etapa de pré-processamento (Seção 5.4.1) não são amostras isoladas dentro das vizinhanças consideradas e, por tal motivo, seus graus de isolamento são baixos. Então, como a escolha do ε ótimo dentro da curva ROC busca deixar o maior número possível de Verdadeiros Positivos e o menor número possível de Falsos Positivos, é necessário dar um valor baixo para ε .

C4: Correção: Tempo de processamento

O tempo de processamento do método OPF-OD aprimorado para o dataset Nursery é de 33.55 segundos na média. A afirmação de que o método não demora mais do que 30 segundos é válida se considerarmos o desvio padrão de 4.47, o que significa que em alguma das execuções o tempo foi inferior a 30 segundos.

Capítulo 6

Towards Building an Ideal Training Set for Supervised Pattern Classification using Optimum-Path Forest

Prólogo

O artigo incluído neste capítulo, em co-autoria com Pedro J. de Rezende e Alexandre X. Falcão, ambos do Instituto de Computação da Universidade Estadual de Campinas, foi submetido à Revista *Pattern Recognition*.

Este artigo apresenta um novo algoritmo de aprendizado para o Classificador OPF que tem como objetivo construir automaticamente um bom conjunto de treinamento para classificação supervisionada. A importância deste problema está em que é muito difícil escolher um conjunto de treinamento que possa resultar numa boa acurácia na classificação de amostras novas e, ao mesmo tempo, não dispenda um tempo excessivo no treinamento do classificador.

O algoritmo deve escolher, ao longo das iterações, quais são as amostras mais adequadas para formar parte do conjunto de treinamento, tentando ao mesmo tempo escolher um número baixo de amostras. Para realizar tal escolha, é utilizada uma Árvore Geradora Mínima computada no conjunto de treinamento. Além disso, no artigo também são discutidos aspectos importantes do método, como o critério para se determinar o número de iterações necessárias, o tamanho inicial do conjunto de treinamento e a eficácia do algoritmo tanto no aprendizado quanto na classificação final.

Os resultados obtidos são muito interessantes, pois o nosso algoritmo, além de encontrar automaticamente o conjunto de treinamento, é capaz de superar a acurácia do Classificador OPF tradicional, precisando apenas de um pequeno incremento no tempo de treinamento.

Abstract

The effectiveness of supervised pattern classifiers strongly depends on a training set with a fixed number of labeled samples. However, the choice of the most representative samples to form a training set of minimum size is an unsolved problem. In this work, we propose a solution based on the optimum-path forest supervised classifier (OPF) that automatically finds a compact training set as it increases the accuracy of classification on a larger evaluating set. We demonstrate with several datasets that the method can improve accuracy in the classification of unseen samples, when compared against the OPF classifier using its traditional learning process. Moreover, the amount of extra time spent in learning was minimal.

6.1 Introduction

Pattern Recognition is the capacity of classifying unknown samples based on information provided by known samples from their same class [1]. That information can be represented in some space in the form of a feature vector for each sample, obtainable in many different ways, depending on the type of data. In the case of images, for example, we can use the color component values of each pixel or maybe some feature extraction technique to obtain a more detailed representation of the image.

In this work, we focus on supervised classification, i.e., we assume that the labels of all samples in the dataset are known. In this approach, one starts off by splitting the dataset into two subsets: the training and testing sets. The training set is used by the classifier to learn the distribution of the classes in the feature space (learning phase) and the criteria to predict the classes of the samples in the testing set (testing phase). Furthermore, most classifiers use a third set as an evaluating set to improve the classification criteria found during the learning phase.

Some examples of supervised classifiers are the widely-known Artificial Neural Networks, Support Vector Machines and the Optimum-Path Forest Classifier.

Artificial Neural Networks (ANN), a largely used classification method, can mathematically model any kind of classification problem [2] by using a collection of one or more ANNs. This method is preferred for several problems because of its strong theoretical foundations and its good results. However, one of its disadvantages is that it spends an extremely large amount of time in the learning phase in order to converge into a solution that reaches accurate classification. Also, it is very sensitive to the choice of parameters.

Support Vector Machines (SVM) [3] were proposed to solve classification problems with overlapped classes by assuming that the samples are separable in a higher-dimensional space [4]. As a binary classifier, it must use a combination of SVMs to solve multi-class problems [5]. One of its advantages is that it can attain a very high classification rate, sometimes better than other approaches. However, its main deficiency is that too much computational time is spent in obtaining the support vectors (learning phase), because it is highly dependent on the size of the training set. Furthermore, the assumption of class-separability in higher-dimensional space usually does not hold [6].

The Optimum-Path Forest classifier (OPF) [7] is a graph-based technique which models classification problems as optimum-path searches on graphs derived from an adjacency

relation between samples in a given feature space (a complete relation in the case of this paper). Class representatives (prototypes) are chosen among the training samples and used to classify the remaining samples based on costs of paths in the graph. This method has, as an advantage, a very low computational learning cost, since it does not have to optimize parameters as ANN and SVM do. Furthermore, it can deal with multi-class and overlapped-class data since its formulation is based on multiple prototypes that represent the various classes.

Generally, OPF obtains an accuracy as high as SVM's and better than ANN's, while its learning time can remain about 70 times faster than SVM's and 400 times faster than ANN's [7]. Its efficiency, and also other works [8, 9, 10] show that OPF is the best choice for several classification problems, mainly in the case of large datasets, which are common in real-life problems, and likely require large training sets (i.e., sets with thousands of samples).

Regardless the chosen method, it is difficult to decide what are the best elements for the training set and how many of them we must use to obtain good results. Essentially, it is necessary to achieve a balance between the representativity of the training set and the computational time needed to train the classifier. Therefore, it is not hard to realize that the training set plays a major role in the classification process because the efficacy of the testing phase depends on how well it represents the whole dataset. On the other hand, it could be infeasible to use a very large training set (depending on the classification method used, e.g., SVM or ANN), because it could be necessary to solve time consuming optimization problems on the whole training set.

In this work, we propose a method to automatically find a compact training set for supervised pattern classification, without any a priori information on the ideal size of that set. This method is an extension of the traditional OPF's learning algorithm and it aims at choosing throughout the iterations the most representative samples for the training set, as it increases accuracy of classification on a larger evaluating set. These samples are defined as prototypes chosen using Minimum Spanning Trees computed on the training set instances along all iterations. The effectiveness of our algorithm is demonstrated by the improvement in accuracy for the classification of unseen samples (testing set), when compared against the OPF classifier using its traditional learning algorithm, at the cost of a slight amount of extra time.

This paper briefly describes the OPF classifier in Section 6.2, introduces the new method in Section 6.3, discusses the testing methodology and empirical results in Section 6.4 and summarizes conclusions in Section 6.5.

6.2 Optimum-path forest classifier

In this section, we present the basic concepts of the OPF classifier, as described in [7], to introduce the framework we use to propose our algorithm.

The goal of the OPF classifier is to obtain an acyclic graph (a forest) containing all the elements of the dataset, such that it represents a partition of the dataset into several classes. In this forest, each class is represented as one or more optimum-path trees rooted at specially chosen elements called prototypes.

Let the dataset Z be divided into three randomly chosen subsets Z_1 , Z_2 and Z_3 , representing the training, evaluating and testing sets, respectively. Let $\lambda(s)$ be a function that gives the correct class $i \in \{1, \dots, c\}$, for each sample $s \in Z$. For the initialization of the algorithm, let the training set Z_1 be represented as a complete graph whose nodes are the elements in Z_1 , where each sample $s \in Z$ is represented as a feature vector \vec{s} and each edge $\langle s, t \rangle$ is weighted by $d(s, t)$, the distance between s and t .

A path $\pi_t^s = \langle s = t_1, t_2, \dots, t_n = t \rangle$ inside the graph is a sequence of adjacent nodes starting at s and with terminus at t . $\langle s \rangle$ is a trivial path and $\pi_{s'}^s \cdot \langle s', t \rangle$ denotes the concatenation of a path $\pi_{s'}^s$ with an edge $\langle s', t \rangle$. A path π_t^s is considered optimum if, for any other path $\pi_t^{s'}$, $f(\pi_t^s) \leq f(\pi_t^{s'})$, where f is a function that evaluates the cost of the path.

In this work, f is chosen as the f_{max} function, defined as:

$$\begin{aligned} f_{max}(\langle s \rangle) &= H(s) \\ f_{max}(\pi_{s'}^s \cdot \langle s', t \rangle) &= \max\{f_{max}(\pi_{s'}^s), d(s', t)\} \end{aligned} \quad (6.2.1)$$

where $H(s)$ is a handicap value, which has a finite value only for the prototypes. The function f_{max} computes the maximum edge cost along a path.

A set of prototypes $S \subset Z_1$ (see Section 6.2.1), will be responsible for classifying each sample $t \in Z \setminus S$. Denote by an S -path to a sample $t \in Z \setminus S$, an optimum path from some sample in S to t . The union of S -paths for all $t \in Z_1$ forms a forest, called the Optimum Path Forest (OPF). This forest is organized as a set of trees, each of them being composed of a prototype $s \in S$ (the tree's root) and the elements classified by that prototype.

For the computation of the OPF, we must estimate for each $t \in Z_1$ the optimum cost $C(t) = \min_{\forall \pi_t^s \text{ in } \Pi(Z_1, A, t)} \{f_{max}(\pi_t^s)\}$, which determines the optimum path π_t^* for t , starting at some $s \in S$. $\Pi(Z_1, A, t)$ is the set of all paths in (Z_1, A) with terminus at t .

The minimization problem above is solved by the procedure shown in Algorithm 5, which computes the Optimum-Path Forest on (Z_1, A) represented as the predecessor map P (see [7]). $L(s)$ represents the label of $s \in Z_1$.

Algorithm 5 – OPF-ALGORITHM

INPUT: Training set Z_1 , λ -labeled prototypes $S \subset Z_1$.
 OUTPUT: Optimum-Path Forest P (predecessor map), path-cost map C , label map L and a list Z'_1 of the training nodes ordered by their path-cost.
 AUXILIARY: Priority queue Q and cost variable cst .

1. **For each** $s \in Z_1 \setminus S$ **do** $C(s) \leftarrow +\infty$.
2. **For each** $s \in S$ **do**
3. $C(s) \leftarrow 0$, $P(s) \leftarrow nil$
4. $L(s) \leftarrow \lambda(s)$, *insert s into Q .*
5. **While** Q *is not empty* **do**
6. *Remove from Q a sample s with minimum $C(s)$ and insert s in Z'_1 .*
7. **For each** $t \in Z_1$ *such that* $C(t) > C(s)$ **do**
8. *Compute* $cst \leftarrow \max\{C(s), d(s, t)\}$.
9. **If** $cst < C(t)$ **then**

- | | | | | |
|-----|---|---|---|--|
| 10. | [| [| [| If $C(t) \neq +\infty$ then remove t from Q . |
| 11. | [| [| [| P(t) $\leftarrow s$, L(t) $\leftarrow L(s)$ and $C(t) \leftarrow cst$. |
| 12. | [| [| [| Insert t in Q . |

Finally, every $t \in Z_2 \cup Z_3$ will be classified in the learning and testing phases using the forest computed by Algorithm 5 (as described in Section 6.2.2).

6.2.1 Selection of Prototypes

Papa et al. [7] proposed to use a Minimum Spanning Tree (MST) computed on Z_1 to identify the prototypes. Essentially, a prototype has the purpose of avoiding optimum paths composed of samples from different classes, since we want to have elements from a single class in each tree (which is rooted on a prototype of that class). To that end, we simply choose prototypes from different classes that share an edge in the MST, because when the forest is computed using Algorithm 5, each prototype will classify elements of its own class (i.e., the paths will have elements of a single class).

Since the total cost of the edges on an MST is minimum over all other spanning trees, and since f_{max} is used, the MST contains an optimum path for every $t \in Z_1$.

After identifying the prototypes in Z_1 , the Optimum-Path Forest is computed using the samples in Z_1 (Algorithm 5).

6.2.2 Classification of Evaluating Samples

This phase consists of assigning for each $t \in Z_2$ a label $L(t)$ based on the discrete partition defined by the OPF on the elements of Z_1 . In other words, we use the partition of Z_1 and extend it to Z_2 .

For the classification of a node $t \in Z_2$, we consider all possible paths from each $s \in Z_1$ to t and then choose the optimum one π_t^* . This optimum path must be the concatenation of an optimum path which is already part of the OPF (i.e., their elements belong to Z_1) with a new edge which will be added to the OPF, linking t to the existing path. This minimization problem is solved incrementally using the path-cost map $C(\cdot)$ (computed by Algorithm 5 for all $s \in Z_1$) to compute the cost of t as $C(t) = \min_{\forall s \in Z_1} \{\max\{C(s), d(s, t)\}\}$.

Let $s \in Z_1$ be the node that minimizes $C(t)$ and $R(s) \in S$ be its root node, such that $\pi_t^{R(s)}$ is an optimum path. Then, the class $L(t)$ of t will be set to $\lambda(R(s))$.

The role of Z'_1 is to speed up the evaluation of the above optimization, which can halt when $\max\{C(s), d(s, t)\} < C(p)$, for a node p whose position in Z'_1 succeeds the position of s [8].

Furthermore, the procedure presented above is also used to classify the unseen samples of the testing set Z_3 .

6.2.3 Learning Algorithms for OPF

The traditional OPF's learning algorithm, proposed by Papa et al. [7], first computes the OPF on Z_1 , then classifies the elements in Z_2 and swaps misclassified samples by non-prototypes. This process is repeated for a few iterations to improve the samples in Z_1

(and hence the accuracy of the classifier). Nevertheless, this learning algorithm assumes fixed sizes for the training and evaluating sets, whose ideal values are unknown.

Some variants of the traditional OPF were proposed [11, 12] aiming at an improvement of its accuracy. These variants propose the use of a k -NN graph in the feature space instead of a complete graph to represent the elements in Z_1 . To that end, it is necessary to compute the k nearest neighbors of each sample $s \in Z_1$ and then compute a probability density function (PDF) using that adjacency relation. The PDF is used to choose the prototypes instead of using the MST. Unfortunately, this approach also assumes fixed sizes of Z_1 and Z_2 .

Papa et al. also proposed a learning algorithm that performs a pruning procedure in the training set to trim it down [9]. This algorithm starts with fixed-size training and evaluating sets and performs the traditional OPF's learning procedure several times. This is done to discover the irrelevant samples in Z_1 according to their role in the classification of the samples in Z_2 . The main disadvantage of this algorithm is that it spends a large amount of time in the learning phase (several learning processes are performed) and also that it needs the size of Z_1 as a parameter. Moreover, this approach greatly differs from our algorithm, since the former works on reducing the training set while the latter seeks to discover the most suitable samples for it.

6.3 Smart Growth of the Training Set

In this section, we present a novel learning algorithm for the OPF classifier, called the Optimum-Path Forest Classifier with Smart Growth of the Training Set (OPF-SGTS). It is based on the traditional OPF's learning algorithm [7], endowed with the automatic construction of a suitable training set.

The basic idea is to start with a joined training-evaluating set ($Z_{1,2}$), an initial small sampling (Z_1) of that set, and $Z_2 = Z_{1,2} \setminus Z_1$. In each iteration of the algorithm, we move misclassified samples from Z_2 to Z_1 , perform a selection of the useful samples which remain in Z_1 and return the remaining ones to Z_2 . In this way, we control the growth of Z_1 by retaining only the most beneficial samples.

After computing the initial sampling to form the training set, we obtain prototypes, from the MST of this set, which become the roots of the trees of the OPF (the classifier). As we will discuss later, the initial size of Z_1 affects only the convergence time of the algorithm, while the convergence accuracy remains the same. Afterwards, the elements of Z_2 are classified, leading to misclassified samples that are relocated to Z_1 where they become candidates for prototypes. Once the new prototypes are identified, the training set retains only prototypes from the current and previous iterations while the residual samples are moved back to Z_2 . An OPF rooted at the current prototypes is computed by Algorithm 5 and the whole process is repeated until a stopping criterion is reached.

The learning algorithm that implements this procedure is described next. The subsequent sections present the details of the initial sampling, of the selection of the most representative samples that constitute Z_1 , and alternative stopping criteria.

Algorithm 6 – OPF-SGTS-ALGORITHM

- INPUT: Training-evaluating set $Z_{1,2}$
- OUTPUT: Final Training Set Z_1 , represented as: Optimum-Path Forest P (predecessor map), path-cost map C , label map L and a list Z'_1 of the training nodes ordered by their path-cost.
- AUXILIARY: Set S of prototypes from all iterations, set S_c of prototypes from the current iteration and set Z_{error} of errors in Z_2 .
1. Set $S \leftarrow \emptyset$ and $Z_1 \leftarrow \emptyset$.
 2. Compute Z_1 by initial sampling on $Z_{1,2}$ and set $Z_2 \leftarrow Z_{1,2} \setminus Z_1$.
 3. Identify the set S_c of prototypes in Z_1 via MST and set $S \leftarrow S \cup S_c$.
 4. Execute $(P, C, L, Z'_1) \leftarrow \text{Algorithm-5}(Z_1, S_c)$.
 5. **Do**
 6. Classify Z_2 using the classifier computed in Line 4.
 7. Set Z_{error} to the set of misclassified samples from Line 6.
 8. Set $Z_1 \leftarrow Z_1 \cup Z_{error}$ and $Z_2 \leftarrow Z_2 \setminus Z_{error}$.
 9. Identify the set S_i of prototypes in Z_1 via MST and set $S \leftarrow S \cup S_c$.
 10. Set $Z_2 \leftarrow Z_2 \cup \{Z_1 \setminus S\}$ and $Z_1 \leftarrow S$.
 11. Execute $(P, C, L, Z'_1) \leftarrow \text{Algorithm-5}(Z_1, S_c)$.
 12. **Until** Stopping Conditions are reached.
 13. **Return** the forest (P, C, L, Z'_1) rooted in S_c .

Notice that the algorithm presents an important modification in the prototype-selection phase, as compared with the traditional OPF's (Section 6.2.1). In the traditional approach, the MST is computed on Z_1 and then the OPF algorithm is executed to set up the forest using the entire Z_1 . Instead, our algorithm uses the entire Z_1 only for computing the MST and then sets up the forest using only the prototypes (from current and past iterations). Exceptionally, at the beginning of the algorithm, we perform the same procedure as the traditional OPF (lines 3-4), because we do not have prototypes from previous iterations.

6.3.1 Initial Sampling for Z_1

It is important to choose a representative set of samples for the first iteration of the learning phase, since the quality of the first prototypes depends on them. Moreover, we would not like to employ an extra parameter, since the idea is to have a method that automatically finds that set. Hence, to decide on a good size for Z_1 , two alternatives seem natural: 1) choose one sample per class, which would be the minimum possible size or; 2) choose a small sampling of $Z_{1,2}$, which would represent the distribution of the classes in a better way. Let us consider the trade offs of these options.

Using one sample per class has the advantage of not needing an extra parameter and also that the growth of Z_1 will be adapted to the geometry of the classes, in the absence of any assumptions on their shapes. However, we would have more learning errors in the first iterations, leading to a larger Z_1 and, consequently, the MST will take more time to be computed. Furthermore, the random selection of samples would lead to an unstable behavior of the algorithm.

On the other hand, if we use a small sampling of $Z_{1,2}$, we have a better representation of its geometry, which leads to fewer errors in the first iterations, a lower learning time and a more stable accuracy along the iterations. Experimental tests employed to determine an adequate size for this sampling showed that the convergence of the accuracy always happens, regardless of its initial size. However, when this sampling is too small, the learning time grows higher due to misclassified samples, while a much larger sampled set (e.g. 10%) also causes the training time to be high, simply due to the size of the training set. Hence, we opted for 1% of $Z_{1,2}$ as a balance between these two scenarios.

6.3.2 Selecting the most Suitable Samples

The selection procedure of the method consists of choosing the samples that will remain in the training set in each iteration of the algorithm. To optimize the size of Z_1 , we would like to choose only the samples that are really necessary to perform the classification on the unseen samples of Z_3 .

Considering the way the Optimum-Path Forest is computed (Algorithm 5), the samples which are responsible for the partition of the training set in classes are the prototypes. Hence, they are the most suitable samples for the training set, as they are the best representatives for their classes and they prevent optimum-paths from different classes, which leads to an appropriate separation of the classes.

As stated before, the main goal of the learning phase is to achieve a better classifier. Therefore, we would like to improve the accuracy of the classifier using information of the correctness of the classification of the samples in Z_2 . For that, we move the misclassified samples from Z_2 to Z_1 , since those samples are important to represent the distribution of the classes, precisely because they were wrongfully classified.

Nevertheless, a different set S_i of prototypes is computed in the i -th iteration, since the training set changes throughout the iterations, i.e., $S = S_1 \cup S_2 \cup \dots \cup S_T$, for a number T of iterations (which depends on criteria described in Section 6.3.3).

At the end of this process, we will have prototypes extracted from different OPFs and we need to organize them to form the final forest of Z_1 , because it is the classifier that will actually perform the testing phase with the samples in Z_3 . The most relevant prototypes are the ones computed in the last iteration, namely S_T . However, the former prototypes, $S \setminus S_T$, that were useful in past iterations to describe the distribution of the classes must also remain in the training set. Nevertheless, they will form the set $Z_1 \setminus S_T$ (non-prototype samples). In other words, we execute Algorithm 5 for the set S and using S_T as the chosen set of prototypes.

This reorganization of the prototypes in Z_1 and the computation of the final OPF are important since treating all samples in Z_1 as prototypes would turn the optimization process in Section 6.2.2 into a nearest-neighbor classification rule, instead of an OPF classifier.

Furthermore, the choice of the set S of prototypes as the only samples in the training set not only allow for a good representation of the distribution of the classes, as was explained above, but might also imply a significant smaller training set in comparison with the initial $Z_{1,2}$. In other words, if the classes have a low degree of overlapping, a small number of prototypes will be sufficient to separate the classes during the learning

phase, which will give us a training set that is both small and representative. This is a major advantage of our method, because its accuracy will be as high as if we were using the complete $Z_{1,2}$ set (since the prototypes will be the same), but spending much less time in the testing phase.

6.3.3 Defining the Stopping Conditions

One of the main reasons for the success of the method in achieving a good representation of the classes is that we use the errors identified during the classification of Z_2 to provide a feedback for the classifier and improve it. Hence, it is natural to expect that the accuracy improves along the iterations, leading to a smaller number of samples being moved from Z_2 to Z_1 . Due to this fact, the set of prototypes goes through considerable changes in the first iterations and only a little in the final ones. Therefore, an interesting way to determine a stopping criterion is to measure how much the set of prototypes changes from one iteration to the next and, once this variation is not significant, halt the process. An efficient way of measuring this variation is to compute the cardinality of the symmetrical difference, SD_i , between the set of prototypes from iterations i and $i - 1$ and compare this number with the absolute number of prototypes, S_i , of iteration i . If $SD_i < \alpha S_i$, for some threshold $0 < \alpha < 1$, we stop the process.

Furthermore, we also consider that the method should stop if the accuracy is stable, i.e., the accuracies of two consecutive iterations have a very small difference or if the accuracy is high (say, more than 99%).

In summary, we have three possible stopping conditions for Algorithm 6: 1) a small symmetric difference; 2) accuracy stabilization or; 3) high accuracy.



C1
(Pg.73)

6.4 Experiments

In this section, the overall approach of the experiments, the datasets used in the tests, the computational environment and the results are described.

All the experiments reported here were performed on an Intel[®] Xeon[®] 4-core 2.50 GHz processor, with 8 GB of RAM, under Linux Gentoo 2.6.31-r10 and *without* multi-thread execution.

6.4.1 Datasets

To validate the proposed approach, we have chosen datasets obtained from the University of California at Irvine's Machine Learning Repository [13] belonging to different fields of study: "Shuttle" (aerospatial technology), "Gamma Telescope" (astrophysics), "Letter-Recognition" (text processing), "Nursery" (medicine), "Isolet" (text processing), "Multi-Feat" (text processing), "Balance" (experimental psychology) and "Haberman" (medicine). Table 6.1 shows the number of samples, of classes and of attributes of each dataset.

The Gamma dataset was mainly used in the study of several aspects of the OPF-SGTS method, which are addressed in the next sections, and the other datasets were useful in the final tests.

Table 6.1: Datasets used in the tests.

Dataset	Num. Samples	Num. Features	Num. Classes
Shuttle	58 000	9	7
K.R.K.	28 056	6	18
Gamma	19 020	10	2
Letter	14 808	16	26
Nursery	12 960	8	5
Isolet	6238	617	26
Multi-Feat	1988	649	10
Balance	625	4	3
Haberman	306	3	2

The accuracy measure used to evaluate the classifiers is computed as follows [7]:

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

$$E(i) = e_{i,1} + e_{i,2} \quad Acc = 1 - \frac{\sum_{i=1}^c E(i)}{2c}$$

where $FP(i)$ and $FN(i)$ are the number of false positives and false negatives for class i and $Z_3(i)$ is the number of samples from class i which belong to Z_3 .

6.4.2 Experimental methodology

Since our method aims at automatically building a suitable training set, we would like to measure how good is the training set it obtains. To achieve that goal, we compare OPF-SGTS's performance (accuracy on Z_3 , learning time and testing time) against the traditional OPF classifier with three different sizes for the training set: 1) the same size as OPF-SGTS's (OPF-equal); 2) 20% larger (OPF-larger) and; 3) 20% smaller (OPF-smaller). In other words, once we execute our method, we use the size of the final training set thus obtained as a parameter for the three versions of the traditional OPF.

Regarding the number of iterations, in the experiments we execute the OPF-SGTS method with an experimentally determined value for the symmetric difference threshold α and let the method discover the number of iterations needed for the learning phase. Then, the other methods are executed with that same number of iterations.

6.4.3 Stopping conditions

As explained in Section 6.3.3, a very important part of the method regards the number of iterations that are sufficient to obtain a suitable training set. In this section, we present an experiment to show that using the symmetric difference of consecutive prototype sets to decide when to stop the method is a very good option.

The approach consists of executing the learning phase for a number of iterations higher than the number discovered by the symmetric difference approach (e.g., 15 iterations) and verifying the variation of the results after the symmetric difference threshold α is reached. Figure 6.1 shows the results of this experiment, presenting the accuracy on Z_2 , the number of prototypes and the symmetric difference.

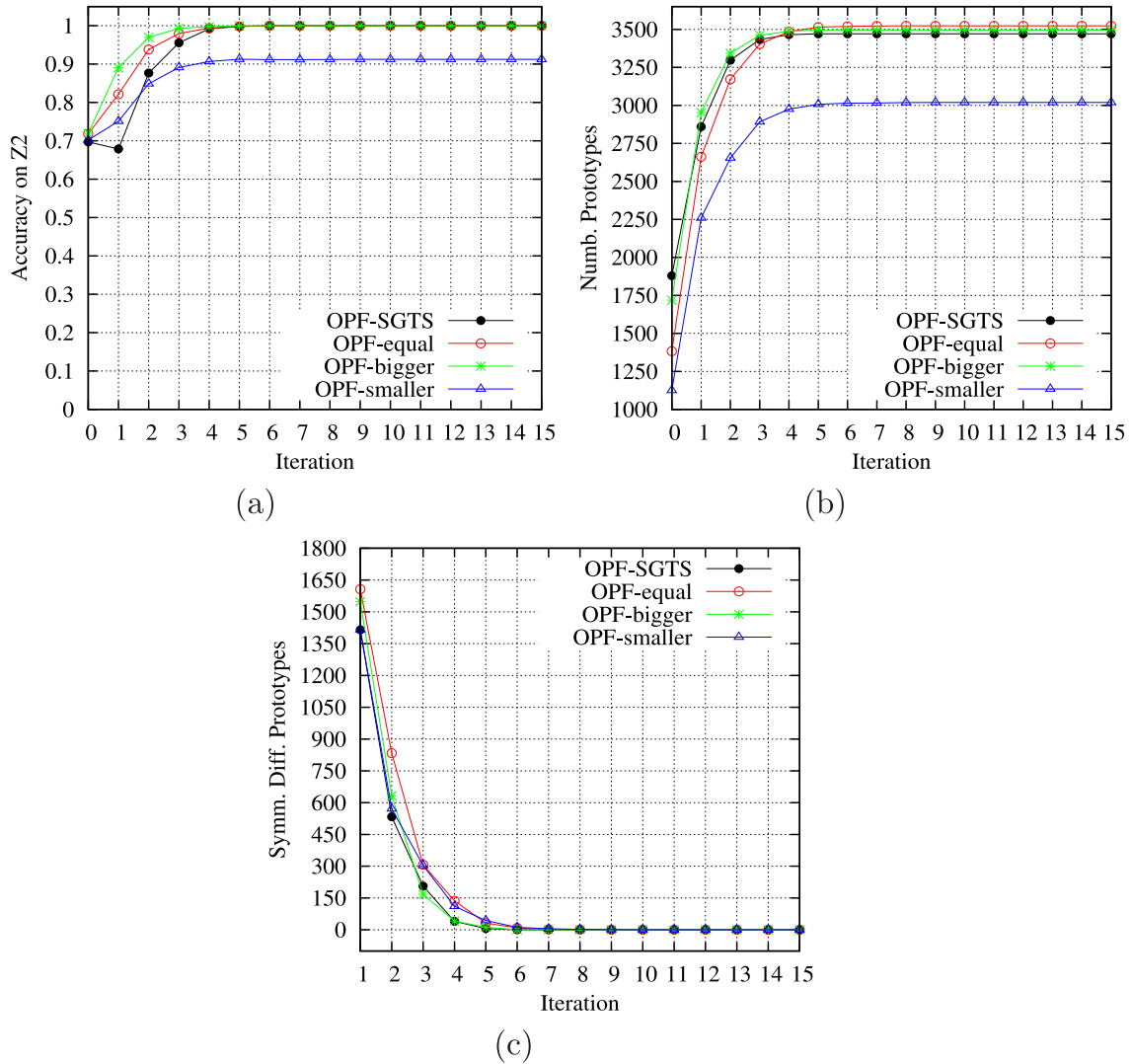


Figure 6.1: Learning process for the four methods using the Gamma dataset. The measures extracted were: (a) accuracy on Z_2 , (b) number of prototypes and (c) symmetric difference. The learning phase was performed with 15 iterations. Here, $|Z_{1,2}| = 50\%$ of $|Z|$.

As it can be seen, after the fifth iteration (which would be the last iteration according to the symmetric difference approach), the accuracy on Z_2 and the number of prototypes remain stable until the end of the process and the symmetric difference remains at zero

after that iteration. This substantiates the decision to stop the process, since after that iteration, there is no gain in performance.

Nevertheless, it might be difficult to choose an appropriate threshold α . If it is too low (close to 0), we would force the method to stop only when there are virtually no variations in the set of prototypes, which could take too much time. On the other hand, if the value is too high, the method could stop in the first couple of iterations, leading to a poor representation of the classes' distribution in the training set. Considering the results in Figure 6.1, we tested values between 0.01 and 0.1 for α , measuring their impact on the final testing phase (i.e., using Z_3). As Figure 6.2 shows, the accuracy on Z_3 does not present significant variations for the different thresholds. However, considering the learning time (monotonically descendant) and the fact that after the fifth iteration in Figure 6.1 (which corresponds to $\alpha = 0.05$), the set of prototypes and the accuracy on Z_2 are stable, it is best to define $\alpha = 0.05$ to achieve a balance between these factors.

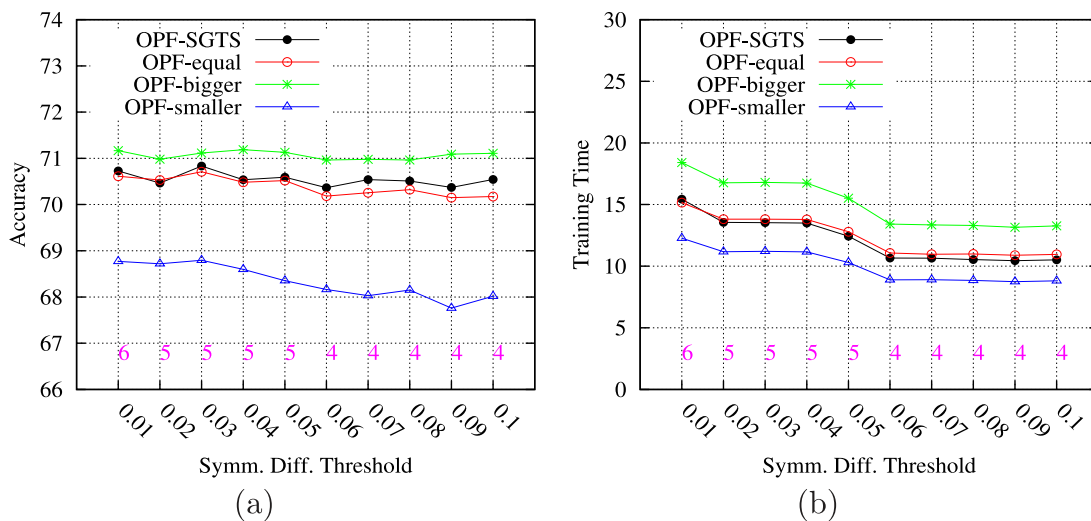


Figure 6.2: Determining a suitable symmetric difference threshold for the OPF-SGTS method. The method was executed with different values of α using the Gamma Dataset, while the other three methods are presented to compare the results. The measures extracted were: (a) accuracy on Z_3 and (b) learning time. The numbers at the bottom are the mean number of iterations discovered by the method for each α . Here, $|Z_{1,2}| = 50\%$ of $|Z|$.

6.4.4 Evaluation of the algorithm on Z_2

Before presenting the accuracy results of the classification on Z_3 , we present experiments regarding the behavior of the method during the learning phase for better understanding. We measured for each iteration: the number of prototypes, the symmetric difference and the accuracy on Z_2 .

As Figure 6.3 shows, the accuracy on Z_2 in the first iteration is the same for the four methods, since there is no information about the errors. Then, the accuracy gets higher

for the other methods, since they have larger training sets (OPF-SGTS starts with a small sampling) but, along the iterations, OPF-SGTS' accuracy gradually improves until it reaches the others' (when OPF-SGTS' training set gets its final size). Regarding the number of prototypes, our method uses more prototypes than OPF-equal at the beginning, but in the last iteration it uses virtually the same number of prototypes. Furthermore, the fact that OPF-SGTS requires almost the same number of prototypes as OPF-bigger, while the latter has more samples to choose its prototypes from, is an evidence of the good performance of the OPF-SGTS. Finally, OPF-SGTS' symmetric difference is smaller than OPF-equal's, which means that OPF-SGTS' set of prototypes is more stable.

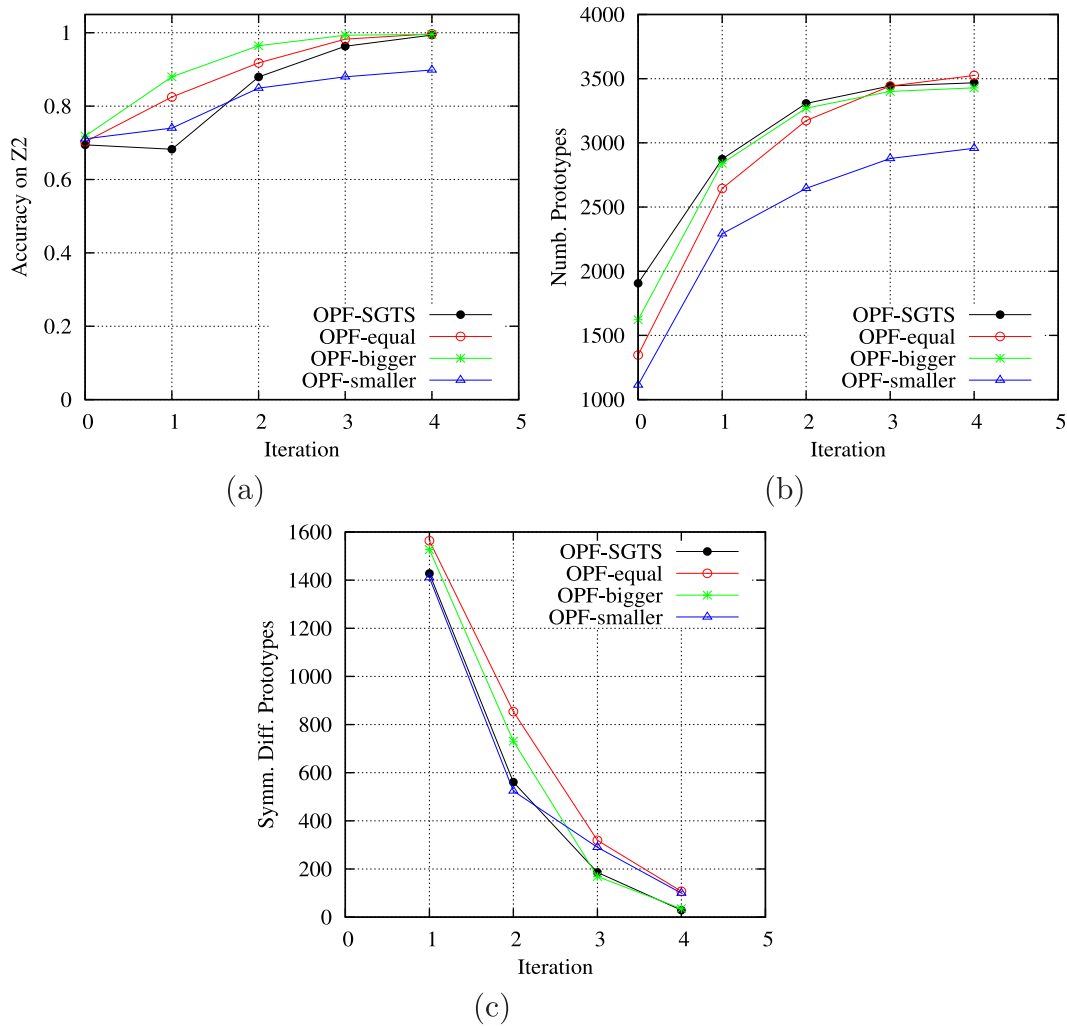


Figure 6.3: Learning process for the four methods using the Gamma dataset. The measures extracted were: (a) accuracy on Z_2 , (b) number of prototypes and (c) symmetric difference for the Gamma dataset. Here, $|Z_{1,2}| = 50\%$ of $|Z|$ and $\alpha = 0.05$. The symmetric difference curve starts from the second iteration because in the first one there are no prototypes.

6.4.5 Evaluation of the algorithm on Z_3

We now consider the accuracy on Z_3 (instead of Z_2) obtained by the method during each of the iterations in the *learning* phase (not during the testing phase!). In other words, we perform a classification of Z_3 during each iteration of the learning phase using the training set obtained in that iteration. This is a meaningful test, because our algorithm aims at building an ideal training set and we want to show that its final version is really the best from all the iterations for classifying the samples in Z_3 .

Figure 6.4 shows these results and, as expected, the final training set is the one which provides for the highest accuracy among all the iterations. For this test, we executed the learning phase during 10 iterations (i.e., without considering the symmetric difference threshold) just to have a better idea of the method's performance.

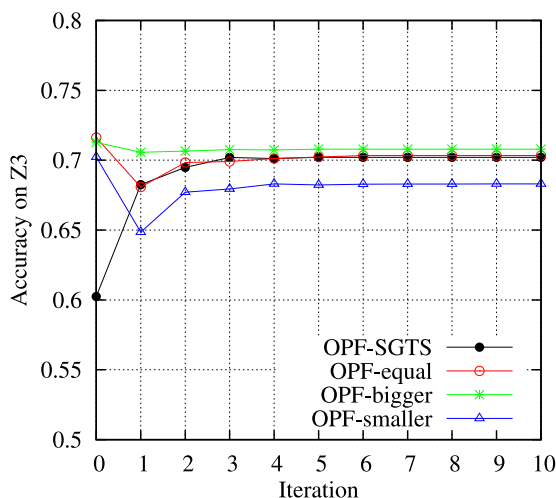


Figure 6.4: Learning process for the four methods using the Gamma dataset and measuring the accuracy on Z_3 (instead of Z_2). Here, $|Z_{1,2}| = 50\%$ of $|Z|$ and $\alpha = 0.05$.

6.4.6 Final results

In this section, we present the results of the method during the testing phase applied to all nine datasets. The process was executed 15 times for each dataset, for $|Z_{1,2}| = 50\%$ of Z and using the symmetric difference approach with $\alpha = 0.05$.

Table 6.2 presents the accuracy on Z_3 for the four methods. As it can be seen, OPF-SGTS has the highest accuracy on seven datasets while OPF-larger, despite its lagniappe, is best for only two of them.

Table 6.3 presents the learning time for the four methods. In almost all of the datasets, our algorithm only presents a slight increase in learning time, which is justified by the computation of an MST on a larger set, Z_1 , in each iteration of the learning process. However, it is important to highlight that our method starts off with a very small sampling from $Z_{1,2}$ and, along the iterations, it determines the most suitable samples for the training set. The size of Z_1 , so determined, is passed on to OPF as a parameter, which then samples

Table 6.2: Accuracy on Z_3 (mean \pm standard deviation) for OPF-SGTS, OPF-equal, OPF-larger and OPF-smaller. Best values are in bold.

Dataset	Accuracy on Z_3			
	OPF-SGTS	OPF-equal	OPF-larger	OPF-smaller
Shuttle	92.39 \pm 2.59	91.48 \pm 4.60	91.66 \pm 3.12	88.83 \pm 5.98
K.R.K.	72.84 \pm 0.67	72.79 \pm 0.38	72.82 \pm 0.65	72.69 \pm 0.47
Gamma	70.67 \pm 0.28	70.46 \pm 0.48	71.09 \pm 0.47	68.61 \pm 0.48
Letter	93.96 \pm 0.26	92.64 \pm 0.41	93.85 \pm 0.20	91.43 \pm 0.57
Nursery	90.63 \pm 3.99	88.80 \pm 4.83	87.95 \pm 4.84	87.23 \pm 4.79
Isolet	90.05 \pm 0.42	89.49 \pm 0.37	90.37 \pm 0.31	88.69 \pm 0.29
M-Feat	94.95 \pm 0.57	94.06 \pm 0.83	94.82 \pm 0.67	93.18 \pm 0.87
Balance	68.61 \pm 1.82	67.44 \pm 1.83	68.30 \pm 2.02	68.25 \pm 2.36
Haberman	57.22 \pm 4.48	56.09 \pm 3.74	57.01 \pm 3.83	56.08 \pm 2.85

off a training set of that size to be refined along its iterations. Nevertheless, OPF-SGTS attains higher accuracy than OPF, even with this considerable advantage for the latter.

Table 6.3: Learning time in seconds (mean \pm standard deviation) for OPF-SGTS, OPF-equal, OPF-larger and OPF-smaller.

Dataset	Learning Time			
	OPF-SGTS	OPF-equal	OPF-larger	OPF-smaller
Shuttle	8.80 \pm 1.12	1.46 \pm 0.21	1.71 \pm 0.24	1.18 \pm 0.16
K.R.K.	112.16 \pm 0.60	80.834 \pm 6.22	101.86 \pm 2.47	65.75 \pm 3.37
Gamma	12.18 \pm 1.09	13.163 \pm 1.14	15.97 \pm 1.40	10.63 \pm 0.93
Letter	5.00 \pm 0.10	4.09 \pm 0.06	4.85 \pm 0.07	3.28 \pm 0.04
Nursery	8.98 \pm 0.43	7.84 \pm 0.34	9.14 \pm 0.43	6.37 \pm 0.29
Isolet	8.8 \pm 0.62	8.23 \pm 0.59	9.83 \pm 0.63	6.50 \pm 0.40
M-Feat	0.59 \pm 0.08	0.49 \pm 0.07	0.57 \pm 0.08	0.39 \pm 0.06
Balance	0.007 \pm 0.005	0.005 \pm 0.005	0.008 \pm 0.005	0.003 \pm 0.004
Haberman	0.001 \pm 0.000	0.001 \pm 0.000	0.001 \pm 0.000	0.001 \pm 0.000

Regarding the time needed to perform the test phase, it is the same for OPF and OPF-SGTS since their training set have the same size. Similarly, OPF-larger and OPF-smaller achieve proportionally higher and lower testing times.

Finally, Table 6.4 shows some useful statistics about the learning phase of OPF-SGTS and OPF-equal, namely, the final size of Z_1 , percentage of prototypes and non-prototypes in Z_1 . In the majority of the datasets, the final size of the training set obtained by OPF-SGTS was between 20% and 50% of the initial $Z_{1,2}$ (which was half of Z). This is an



C4
(Pg.73)



C5
(Pg.74)

acceptable size, since it means that by using between 10% and 25% of Z for the training set, we achieve accuracies between 70% and 95% on Z_3 . Moreover, in the case of the Shuttle dataset, the final Z_1 was less than 1% of Z , and the accuracy on Z_3 was 92%, which is an outstanding performance.

The number of prototypes in Z_1 was very similar for OPF-SGTS and OPF-equal. However, it is important to emphasize that our method was very stable along all 15 executions, i.e., a low standard deviation in the number of prototypes in Table 6.4, while OPF-equal achieved very dissimilar results in each execution, i.e., higher standard deviation. This is an important observation because, with a stabler number of prototypes, the overall performance of the method itself will be equally unwavering.

Table 6.4: Learning statistics for OPF-SGTS and OPF-equal. Here, $|Z_{1,2}|$ and $|Final Z_1|$ are expressed in number of samples, while $Prot(Z_1)$ and $Non.Prot(Z_1)$ are percentages from $|Z_1|$. The values are presented as (mean \pm standard deviation).

Dataset	$ Z_{1,2} $	$ Final Z_1 $	OPF-SGTS		OPF-equal	
			$Prot(Z_1)$ [%]	$Non.Prot(Z_1)$ [%]	$Prot(Z_1)$ [%]	$Non.Prot(Z_1)$ [%]
Shuttle	28 998	204 \pm 11	89.86 \pm 1.79	10.15 \pm 1.79	32.61 \pm 8.48	67.39 \pm 8.48
K.R.K.	14 024	10 790 \pm 84	78.88 \pm 0.52	21.12 \pm 0.52	85.30 \pm 8.82	14.70 \pm 8.82
Gamma	9395	3800 \pm 42	91.05 \pm 0.50	8.95 \pm 0.50	89.09 \pm 9.57	10.91 \pm 9.57
Letter	7396	1947 \pm 25	89.20 \pm 0.75	10.80 \pm 0.75	85.92 \pm 5.57	14.08 \pm 5.57
Nursery	6480	1887 \pm 58	78.91 \pm 1.40	21.10 \pm 1.40	85.23 \pm 7.05	14.77 \pm 7.05
Isolet	3119	1015 \pm 15	91.11 \pm 0.93	8.89 \pm 0.93	90.23 \pm 5.88	9.77 \pm 5.88
M-Feat	994	198 \pm 12	88.57 \pm 2.48	11.43 \pm 2.48	83.88 \pm 5.45	16.12 \pm 5.45
Balance	312	114 \pm 6	89.56 \pm 2.96	10.44 \pm 2.96	87.64 \pm 7.08	12.36 \pm 7.08
Haberman	152	78 \pm 7	92.68 \pm 2.46	7.32 \pm 2.46	88.99 \pm 8.54	11.01 \pm 8.54

6.5 Conclusions

In this work, we have presented a novel learning algorithm that automatically finds a nearly ideal training set for supervised pattern classification, consisting of a very small and well chosen set of samples that allows for a suitable representation of the original set. This algorithm is based on the Optimum-Path Forest Supervised Classifier and aims at choosing, through a number of iterations, the most useful samples for the training set, ensuring a good representation of the whole dataset without requiring an inviable learning time. The most suitable samples are chosen as the prototypes in the training set, as they are the best representatives for their classes and they prevent optimum-paths from different classes, which leads to an appropriate partition of the feature space.

The choice of prototypes may imply a significant reduction of the initial learning set, leading to a very small training set, as was perceived in the case of the Shuttle Dataset, see the results on Table 6.4.

The proposed approach to automatically determine the number of iterations during the learning phase, which is based on the size of the symmetric difference between training sets from consecutive iterations, was found to be effective, as demonstrated in the experiments presented in Section 6.4.3.

The efficacy of our algorithm is confirmed by the improvement in accuracy obtained against the traditional OPF classifier for a large majority of the datasets used in the tests (Table 6.2), while bearing only a slight amount of extra time (Table 6.3) justified by the computation of an MST on a larger training set in each iteration. Furthermore, our algorithm has a stabler performance when compared to the traditional OPF classifier, in regard to the number of prototypes on the training set, as it can be seen in Table 6.4. This stability also brings a better overall dependability to the method.

Acknowledgments

César Castelo-Fernández was supported by CAPES Grant # 01-P-04388/2010. Pedro J. de Rezende was partially supported by CNPq Grants # 472504/2007-0, 483177/2009-1, 473867/2010-9 and FAPESP Grant #07/52015-0 and a Grant from FAEPEX|UNICAMP. Alexandre X. Falcão was partially supported by CNPq Grants #481556/2009-5 (ARPIS), 303673/2010-9 and FAPESP Grant #07/52015-0. The authors would also like to thank João Paulo Papa from the State University of São Paulo (UNESP), for discussions on variations of the OPF learning algorithm.

References

- [1] R. Duda, P. Hart, D. Stork, *Pattern Classification*, 2nd Edition, Wiley-Interscience, 2000.
- [2] S. Haykin, *Neural Networks, a comprehensive foundation*, 2nd Edition, Pearson Education, 1999.
- [3] V. Vapnik, An overview of statistical learning theory, *IEEE Transactions on Neural Networks* 10 (5) (1999) 988–999.
- [4] C. Burges, A tutorial on support vector machines for pattern recognition, *Data mining and Knowledge Discovery* 2 (1) (1998) 121–167.
- [5] K. Duan, S. Keerthi, Which is the best multiclass SVM method? an empirical study, *Multiple Classifier Systems* (2005) 278–285.
- [6] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: *Proceedings of the 5th Workshop on Computational Learning Theory*, ACM Press, New York, NY, USA, 1992, pp. 144–152.
- [7] J. Papa, A. Falcão, C. Suzuki, Supervised pattern classification based on optimum-path forest, *International Journal of Imaging Systems and Technology* 19 (2009) 120–131.

- [8] J. P. Papa, F. A. M. Cappabianco, A. X. Falcão, Optimizing optimum-path forest classification for huge datasets, in: *Proceedings of The 20th International Conference on Pattern Recognition*, IEEE Computer Society, 2010, pp. 4162–4165.
- [9] J. P. Papa, A. X. Falcão, G. M. Freitas, A. M. H. Ávila, Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation, *IEEE Geoscience and Remote Sensing Letters* 7 (2) (2010) 396–400.
- [10] L. M. Rocha, F. A. M. Cappabianco, A. X. Falcão, Data clustering as an optimum-path forest problem with applications in image analysis, *Intl. J. of Imaging Systems and Tech.* 19 (2) (2009) 50–68.
- [11] J. P. Papa, A. X. Falcão, A new variant of the optimum-path forest classifier, in: *Proceedings of the 4th International Symposium on Advances in Visual Computing, Lecture Notes on Computer Science*, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 935–944.
- [12] J. P. Papa, A. X. Falcão, A learning algorithm for the optimum-path forest classifier, in: *-Based Representations in Pattern Recognition*, Springer Berlin/Heidelberg, Venice, Italy, 2009, pp. 195–204.
- [13] A. Asuncion, D. Newman, UCI machine learning repository (2007), www.ics.uci.edu/~mllearn/MLRepository.html (Access date: August 30th 2010).

Comentários

C1: Critérios de parada adicionais

Como foi explicado, a proposta para controlar o número de iterações do método é a utilização da cardinalidade da Diferença Simétrica entre os protótipos de iterações consecutivas. O objetivo da utilização dos critérios adicionais (limiares de estabilidade de acurácia e acurácia alta) é apenas para garantir que o método pare em caso de não ser alcançado o critério da Diferença Simétrica. No entanto, em todos os testes realizados, o critério de parada foi a Diferença Simétrica.

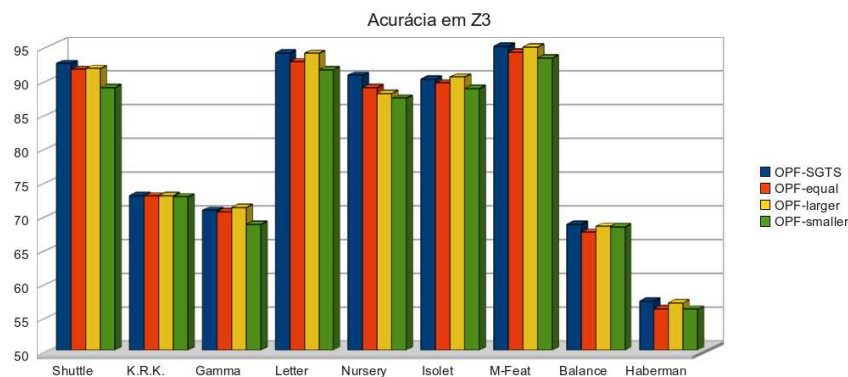
Em relação aos valores desses limiares, não foi necessário testar diferentes possibilidades, justamente pela efetividade da Diferença Simétrica. Então, apenas decidimos exigir uma acurácia muito alta (99%) ou muito estável (diferença de 0.001 entre acurácias de iterações consecutivas).

C2: Medida de acurácia

A medida de acurácia apresentada na Seção 6.4.1 corresponde à avaliação do método na fase de teste, motivo pelo qual se utiliza o conjunto Z_3 . No entanto, as medidas de acurácia nas Seções 6.4.3 e 6.4.4 (identificadas como “*Accuracy on Z_2* ”) se referem à fase de aprendizado e, portanto, foi utilizado o conjunto Z_2 no cômputo dessa medida.

C3: Gráfico dos Resultados

Para se ter uma melhor idéia dos resultados do método OPF-SGTS sobre o restante das bases de dados, apresentam-se os resultados de acurácia da Tabela 6.2 no seguinte gráfico:



C4: Correção: Tempo de teste

O tempo de teste (para o classificador OPF em geral) não depende apenas do tamanho do conjunto de treinamento. De acordo com uma melhoria que foi proposta ao algoritmo de classificação do OPF, não é necessário consultar todas as amostras no processo de classificação, ou seja, o processo pode terminar antes. Como foi explicado na Seção 6.2.2, isso acontece quando se determina que a partir de uma certa amostra não existe nenhuma outra que pode oferecer um melhor custo.

C5: Redução do conjunto do treinamento

Como foi apresentado na Tabela 6.4, o tamanho final do conjunto de treinamento para a base Shuttle foi muito pequeno em comparação com o tamanho do conjunto inicial $Z_{1,2}$, sendo que o Z_1 final é menos de 1% de $Z_{1,2}$.

No entanto, considerando o critério de escolha do método OPF-SGTS, podemos dizer que a razão para tal redução é provavelmente a distribuição das classes no espaço de características, ou seja, é muito provável que as classes sejam linearmente separáveis. Portanto, a quantidade de protótipos necessários para separar as classes é muito baixa, e como o método OPF-SGTS deixa em Z_1 apenas os protótipos, o resultado final é um conjunto de treinamento muito pequeno.

Capítulo 7

Trabalhos futuros

Dentre os principais trabalhos futuros que surgiram ao longo do desenvolvimento desta dissertação, podemos destacar os seguintes:

- Desenvolvimento de um classificador híbrido entre o classificador OPF e algum outro classificador supervisionado como SVM, Redes Neurais, k -NN, etc, utilizando o algoritmo de aprendizado proposto para a construção automática de um conjunto de treinamento para classificação supervisionada (OPF-SGTS). A razão para se pensar nesta proposta é que o algoritmo OPF-SGTS é muito flexível em relação aos algoritmos que podem ser utilizados nas etapas que o formam.

Depois de ser feita a amostragem inicial do conjunto de treinamento, podemos utilizar qualquer algoritmo para fazer o treinamento com Z_1 e posteriormente classificar as amostras em Z_2 . Essencialmente, apenas é necessário saber quais amostras foram classificadas incorretamente, para serem levadas a Z_1 e, posteriormente, computar a MST para definir quais são os protótipos dessa iteração. Portanto, desse modo também é possível deixar os protótipos de todas as iterações em Z_1 e construir automaticamente um bom conjunto de treinamento.

Um aspecto interessante dessa proposta é que podemos aproveitar as vantagens individuais dos diferentes algoritmos de classificação. Por exemplo, se usarmos OPF e SVM, podemos realizar o treinamento com OPF (que é muito mais rápido que o SVM) e a classificação com SVM (que pode ser mais rápida que a do OPF, dependendo do número de vetores de suporte).

- Uma das motivações para desenvolvermos o algoritmo OPF-SGTS foi a sua aplicação em segmentação interativa de imagens naturais, pois um dos principais desafios nessa área é a escolha das amostras de treinamento a partir das marcações feitas por um usuário. Neste sentido, um trabalho a ser realizado no futuro seria a aplicação do nosso algoritmo no software USIS [33], desenvolvido no Laboratório de Informática Visual (LIV-IC-Unicamp), cuja finalidade é a investigação de técnicas de segmentação de imagens naturais de grandes dimensões (e.g., imagens de 8mpx). As marcações feitas pelo usuário neste tipo de imagens estão formadas por vários

milhares de pixels e, devido ao caráter interativo do sistema, o tempo de resposta deve ser muito baixo.

Atualmente, o conjunto de treinamento é escolhido através de uma amostragem de tamanho fixo dos marcadores feitos pelo usuário. Porém, não é difícil perceber que esta amostragem pode não representar adequadamente os marcadores, pois o conjunto inicial deve ser muito pequeno para se conseguir fazer a segmentação em tempo real. Portanto, resulta muito interessante a aplicação do nosso algoritmo OPF-SGTS para substituir a amostragem aleatória feita atualmente no software. Com essa mudança conseguiríamos escolher um conjunto que represente mais adequadamente os marcadores e, como foi explicado no artigo, o tempo necessário seria apenas um pouco maior ao tempo de aprendizado do OPF tradicional. Além disso, poderia ser proposto algum critério que permita estabelecer um tamanho máximo do conjunto de treinamento construído pelo algoritmo, para atender as restrições de tempo de resposta do sistema.

- A metodologia proposta no segundo artigo para a detecção dos outliers no conjunto de treinamento possui propriedades interessantes que podem ser aplicadas na resolução de outros problemas. Especificamente, a Densidade Relativa computada para cada amostra através dos falsos positivos/negativos e verdadeiros positivos/negativos poderia substituir o cálculo da Densidade de Probabilidade em algoritmos de Classificação Não-Supervisionada (clustering), como o Clustering por Floresta de Caminhos Ótimos [30, 6].

Uma grande vantagem da aplicação do nosso método seria a redução no tempo de treinamento, pois, no referido algoritmo, a etapa mais demorada é o cálculo dos vizinhos mais próximos de cada amostra. Essa relação de vizinhança é necessária para computar a PDF do conjunto de treinamento, que posteriormente é utilizada para definir as raízes da floresta. Devido ao fato da Densidade Relativa ser uma aproximação da densidade de probabilidade, é possível que exista uma perda na acurácia do classificador, porém, considerando os resultados do algoritmo proposto, espera-se que esta perda não seja muito significativa.

Capítulo 8

Considerações Finais

Nesta dissertação, foi tratado o problema de classificação supervisionada de padrões, que é amplamente estudado na literatura pela sua importância em diversas áreas do conhecimento, como medicina, biologia, química, física, bioinformática, etc. Especificamente, foram propostos três novos algoritmos de aprendizado para o Classificador baseado em Florestas de Caminhos Ótimos (OPF), que vem sendo estudado nos últimos anos devido à eficiência e eficácia que vem demonstrando na resolução de diversos problemas.

O primeiro algoritmo proposto visa à melhoria da acurácia do OPF tradicional através da detecção de outliers no conjunto de treinamento. Esses outliers são detectados medindo-se os erros e acertos obtidos por eles na classificação das amostras de avaliação. A contribuição da nossa proposta é comprovada pela melhoria em acurácia em relação ao OPF tradicional, o que significa que é importante detectar os outliers presentes em uma base de dados, pois a sua ausência traz melhorias na etapa de classificação.

Posteriormente, foi proposto um aprimoramento na forma de se detectarem os outliers no conjunto de treinamento e é apresentado um estudo mais completo sobre o comportamento do algoritmo com diversas bases de dados. As mudanças feitas no algoritmo de aprendizado permitem obter uma melhor taxa de detecção de outliers e, além disso, a forma como se determina as amostras que são outliers permite explorar propriedades interessantes do método. Uma destas propriedades é a Densidade Relativa computada para cada amostra.

A Densidade Relativa é um modo de explorar a vizinhança de uma amostra sem a necessidade de computá-la explicitamente, pois obtém-se um valor aproximado da densidade de probabilidade de uma amostra (PDF) através da quantidade de acertos e erros obtidos por ela. Graças a essa observação, seria possível utilizar o nosso método para resolver outros problemas que requerem o cálculo da PDF, como a Classificação Não-Supervisionada de padrões, onde uma das etapas mais custosas é o cálculo da PDF, pois requer computar a vizinhança de cada amostra.

A metodologia de teste escolhida para a avaliação do segundo algoritmo permite utilizá-lo para resolver um problema muito interessante em reconhecimento de padrões, que é a Detecção de Erros na rotulação manual ou automática de bases de dados.

Finalmente, foi proposto um algoritmo que permite resolver satisfatoriamente um problema muito difícil em reconhecimento de padrões, que é a escolha de um conjunto de

treinamento adequado para realizar tarefas de classificação supervisionada de padrões.

Os resultados mostraram que o algoritmo proposto é capaz de construir automaticamente um conjunto de treinamento com tamanho muito pequeno que permita obter uma acurácia igual ou melhor que a obtida pelo classificador OPF tradicional.

A forma como está organizado o algoritmo oferece a possibilidade de se utilizar diversos algoritmos de aprendizado e classificação, permitindo a criação de classificadores híbridos que possam aproveitar as vantagens individuais dos classificadores envolvidos.

Os resultados satisfatórios do algoritmo proposto sugerem a sua utilização em sistemas de segmentação interativa de imagens com milhões de pixels, onde é muito difícil escolher, a partir dos marcadores feitos pelo usuário, quais são as amostras mais adequadas para o conjunto de treinamento.

Referências Bibliográficas

- [1] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763.
- [2] 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.
- [3] A.J. Broder. Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23(1-2):171–178, 1990.
- [4] A.E. Bryson and Y.C. Ho. *Applied optimal control: optimization, estimation, and control*. Blaisdell Publishing Company, 1969.
- [5] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and Knowledge Discovery*, 2(1):121–167, 1998.
- [6] 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.
- [7] C. Castelo-Fernández, P.J. de Rezende, A.X. Falcão, and J.P. Papa. Improving the accuracy of the optimum-path forest supervised classifier for large databases. In *15th Iberoamerican Congress on Pattern Recognition (CIARP)*, volume LNCS 6419, pages 467–475, Springer, Nov 2010.
- [8] C. Castelo-Fernández, P.J. de Rezende, A.X. Falcão, and J.P. Papa. A novel outlier detection method based on the optimum-path forest supervised classifier. *Submitted*, Jan 2011.
- [9] C. Castelo-Fernández, P.J. de Rezende, and A.X. Falcão. Towards building an ideal training set for supervised pattern classification using optimum-path forest. *Submitted*, Mar 2011.
- [10] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- [11] B. Dasarthy. *Nearest neighbor Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

- [12] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*. Applications of mathematics. Springer, 1996.
- [13] A. Djouadi and E. Bouktache. A fast algorithm for the nearest-neighbor classifier. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):277–282, 1997.
- [14] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000.
- [15] 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.
- [16] R. Fletcher. *Practical Methods of Optimization, 2nd Edition*. John Wiley and Sons, Inc., 1987.
- [17] K. Fukunaga and P.M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24(7):750–753, 1975.
- [18] 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.
- [19] S. Haykin. *Neural Networks: a comprehensive foundation*. Prentice Hall, 1994.
- [20] 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.
- [21] L.J. Hubert. Some applications of graph theory to clustering. *Psychometrika*, 39(3):283–309, 1974.
- [22] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- [23] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- [24] 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.
- [25] 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.

- [26] S.A. Nene and S.K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003, 1997.
- [27] J.P. Papa. *Classificação Supervisionada de Padrões Utilizando Floresta de Caminhos Ótimos*. PhD thesis, Instituto de Computação, Universidade Estadual de Campinas, 2009.
- [28] J.P. Papa and A.X. Falcão. On the training patterns pruning for optimum-path forest. In *Proceedings of the 15th International Conference on Image Analysis and Processing*, pages 259–268, Berlin/Heidelberg, 2009. Springer-Verlag.
- [29] J.P. Papa, A.X. Falcão, and C.T.N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19:120–131, 2009.
- [30] L.M. Rocha. *Agrupamento de Dados por Floresta de Caminhos Ótimos*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 2008.
- [31] B. Scholkopf and A.J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, USA, 2002.
- [32] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.
- [33] T.V. Spina and A.X. Falcão. USIS: user-steered image segmentation framework, URL <http://code.google.com/p/ift-demo/> (data de acesso: março de 2011), 2010.
- [34] S. Theodoris and K. Koutroumbas. *Pattern Recognition*. Elsevier Science, 1 edition, 1999.
- [35] V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.
- [36] 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.
- [37] M.S. Yang and H.S. Tsai. A gaussian kernel-based fuzzy c-means algorithm with a spatial bias correction. *Pattern Recognition Letters*, 2008.
- [38] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1):68–86, Jan. 1971.
- [39] 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.

- [40] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.